



Technical Report

**Final Report: Advancing SensorNet
Technologies to Monitor Trusted Corridors**

University of Kansas
Information and Telecommunication Technology Center
D.D. Deavours, J.B. Evans, V.S. Frost, G.J. Minden,
D.W. Petr, D. DePardo, E. Komp, L. Searl,
S. Aroor, D.T. Fokum, M. Kuehnhausen, P. Mani,
S. Muralidharan, A.N. Oguna, and M. Zeets

EDS/HP Enterprise
M. Gatewood, S. Hill, L. Sackman, J. Spector, J. Strand,
T. Terrell and J. Walther

ITTC-FY2010-TR-41420-26
June 2010

Project Sponsor:
Oak Ridge National Laboratory (ORNL)
Award Number 4000043403

Abstract

This effort demonstrated an integrated data-oriented methodology that can increase efficiency and security by monitoring cargo movements along a trusted corridor, and especially rail facilities. This was achieved by developing, analyzing, and evaluating the Transportation Security SensorNet (TSSN) for rail transportation, and by demonstrating its feasibility in two field trials. These results have laid the foundation for increased private sector efficiency through close collaboration with Kansas City SmartPort's Trade Data Exchange system, and the increased security benefits law enforcement and national security. The project culminated with a "long haul test" that monitored rail-based cargo in central Mexico, combining real-time sensing and trade data.

Table of Contents

Abstract.....	1
Table of Contents.....	2
List of Figures.....	2
List of Tables.....	3
1. Introduction.....	4
2. Background.....	4
3. Completed Tasks.....	5
3.1. Task 1: Intermodal Technology Proof of Concept and Integration of the SmartPort Trade Data Exchange architecture and SensorNet technologies.....	5
3.2. Task 2: System architecture development, design, measurement and modeling for the SmartPort intelligent transportation systems, with a focus on identification of bottlenecks and scaling issues.....	9
3.2.1. Transportation Security SensorNet Architecture.....	9
3.2.2. Transportation Security SensorNet Measurement and Modeling.....	10
3.3. Task 3: Data integration and processing, e.g., controlling the storing and access of information in the SmartPort data clearinghouse.....	15
3.4. Task 4: Communications technologies to enable continuous monitoring.....	16
3.5. Task 5: RFID applications.....	17
4. Description of Student Activities.....	18
5. Conclusions.....	19
6. References.....	20
List of Appendices.....	23

List of Figures

Figure 3: Image of the short haul test. The small box attached to the top-front of the locomotive contains the external electronics box (described below).....	7
Figure 4: Electronics box outside of locomotive during long haul test.....	7
Figure 5: Placement of sensor on container.....	7
Figure 6: Train passing through central Mexico during long haul test.....	8
Figure 7: Partial route of long haul test northbound from San Louis Potosi.....	8
Figure 9: High-level architecture of the TSSN.....	9
Figure 10: Architecture of the MRN Sensor Node.....	10
Figure 11: Architecture of the Virtual Network Operations Center (VNOC).....	12
Figure 12: Typical sequence of events that take place from a tamper event to notification of a decision-maker.....	13
Figure 13: Optimal sensor locations (a), where Unit ID 0 is the locomotive and slot represents elevation of double-stacked containers, and visibility vs. cost (b) obtained by varying the number of sensors.....	13
Figure 14: Cost of track-side system varying the required response time (a) and both the trackside reader cost and response deadline (b).....	14
Figure 15: Comparison of train-mounted and trackside communication systems.....	14
Figure 16: Security architecture used between TSSN and TDE.....	15
Figure 17: Distributed shared queue model of message exchanges for unreliable communications channel.....	16
Figure 18: Measured results of message dwell times during the long haul trial.....	17

List of Tables

Table I: Summary of time statistics for decision-maker notification.	13
Table II: List of students and their participation on the project.	18

1. Introduction

Exports from Asia have increased creating bottlenecks at key US ports. Conterminously, a Kansas City group, known as SmartPort, recognized the strategic position of Kansas City and has actively worked to expand its role in distribution to increase traffic through the Kansas City area. SmartPort is developing a US export capability and has the only Mexican Customs clearance capability that is not at the border. One of the key goals to expanding this program is the creation of trusted corridors. In support of that goal, the purpose of this research effort has been to develop and integrate systems that provide the ability to track and monitor the security of cargo in transit. This tracking serves two purposes: it provides greater visibility to stakeholders, and it provides sensor-based security to enable corridors to be trusted.

These goals were met through the execution of five tasks.

1. Integration of a distributed sensor system, known as SensorNet, with the SmartPort architecture.
2. Development of the system engineering models and approaches required to support the design optimization and lifecycle operation to the SensorNet-enabled trusted corridor.
3. Development of information systems required for the SmartPort data clearinghouse.
4. Development of the communication system required for monitoring cargo in transit.
5. Examination of the role of RFID in trusted corridors.

The project has come to a successful completion with numerous publications describing the results of the research activities. The following sections contain a summary to those activities and references to details where appropriate.

2. Background

The US economic security is based in part by the efficient transportation of goods. The key ports of entry on the US West Coast are Los Angeles/Long Beach, Seattle/Tacoma, Oakland, and Portland, with LA/Long Beach being the largest. Recent events on the west coast, e.g., the Longshoreman's Strike, Union Pacific track problems, noise and environmental concerns, limitations of the Alameda corridor, etc., highlight the vulnerability of that port. Further, any disaster, including terrorist attacks, will hypothetically shut down the targeted port. As a consequence, a number of companies are developing backup plans utilizing other ports. Some companies are moving their businesses to less busy ports; others are now splitting their cargos between ports. A number of companies are looking to the West-coast Mexican ports for relief. The three principal West-coast Mexican ports are Ensenada, Manzanillo, and Lazaro Cardenas. Kansas City-based Kansas City Southern Railway has the ability to land cargo at Lazaro Cardenas and carry it all the way to the center of the US, i.e., terminating in Kansas City. In a related move, Mexican Customs recognized the strategic location of Kansas City and is now building its first Customs office outside Mexico. Thus, an integrated SensorNet-based system is useful to secure these trusted corridors.

Additionally, industry associations including KC SmartPort have indicated the industry's need for visibility into freight and cargo movement. There are intermodal "black holes" when freight changes hands across modes and carriers. Visibility will only be possible through the integration of carrier, shipper, broker, importer, exporter, and forwarder information. Currently, industry is demonstrating that it is possible to integrate disparate transportation information. The SmartPort Trade Data Exchange (TDE) was contemporaneously developed to address this need, which has laid the foundation for large-scale information integration. The successful technology demonstration of SensorNet's Transportation Security SensorNet (TSSN) has demonstrated the successful integration of the TSSN and TDE. This integration has enhanced value to SmartPort's TDE by providing greater visibility into these "black holes" and securing trade lanes.

This effort focused on use cases centered on monitoring and tracking containers with the goals of proving that a container breach did not occur during the stakeholder's custody and providing time and location of a container intrusion to enable the stakeholder's response and reduce successful intrusions. These were selected in collaboration with our rail stakeholder, Kansas City Southern (KCS). Figure 1 shows the test environment. Figure 2 shows the selected proof of concept technologies. The associated equipment has been acquired and integrated into a complete system. This includes servers to host the TDE at HP (formerly EDS) in Overland Park, KS, and the seals, tags, reader, vehicle mounted TSSN collector (laptop), and a virtual network operations center (VNOC) functionality at ITTC/KU in Lawrence, KS. Experiments have been conducted with the Hi-G-Tek seals, tags, reader, and software developed to integrate them into this system. Initial communications and interactions have been established between the TDE at EDS and the VNOC in the TSSN at ITTC/KU.

3. Completed Tasks

The project was divided into five tasks. Below, we discuss each of the tasks, give summary of the results, and reference the corresponding appendix for more detailed information.

3.1. Task 1: Intermodal Technology Proof of Concept and Integration of the SmartPort Trade Data Exchange architecture and SensorNet technologies

This task produced and demonstrated an integrated SensorNet Transportation Security SensorNet (TSSN) and SmartPort Trade Data Exchange (TDE) architecture and a field sensing prototype for intermodal transport. The system architecture is described in greater detail in the following section (Section 3.2). This task was performed in a number of phases, including: 1) truck-based trials, 2) a "short haul" trial, and 3) a "long haul" trial. Each of those is described briefly below.

We performed a number of trials using trucks to simulate a number of scenarios without interfering with rail operations. Truck trials were used to validate the technology, take measurements, and test boundary conditions.

In January of 2009, we performed an integrated test on the TDE and TSSN to rail-based monitoring of containers in a short, cross-city rail trial with the invaluable assistance of Kansas City Southern, a key stakeholder. Figure 3 shows one of the images of the instrumented train in progress. The results were successful and are documented in [11, 16].

The third phase involved monitoring of a cargo container from San Luis Potosi in the center of Mexico (originally planned for Lazaro Cardenas, but changed due to scheduling difficulties with key partners) through Nuevo Laredo and into the US. The results are summarized in numerous publications, including [10, 15, 16, 17, 18, 20, 21, 22]. Figure 4 shows the external instrumentation of the locomotive, and Figure 5 shows one of the monitored cargo containers with an attached seal. Figure 6 was a picture taken from the train as it was moving through central Mexico, and various GPS measurements taken from the route are shown in Figure 7.

Figure 8 shows a sample email alert that was sent when the TSSN detected an alarm, in this case, an Open event (one seal had been intentionally opened). An associated GPS reading was taken, and the TSSN sent the alarm with a Google Maps link. Before the alarm was sent, the TSSN queried the TDE for shipment data, which is also presented in the alarm. This particular event took place where there was virtually no cell coverage, and thus relied on satellite communication for communication between the train and a virtual network operations center, which for this demonstration, was on University of Kansas campus in Lawrence, KS. The TDE database was in the HP/EDS facilities in Overland Park, KS.

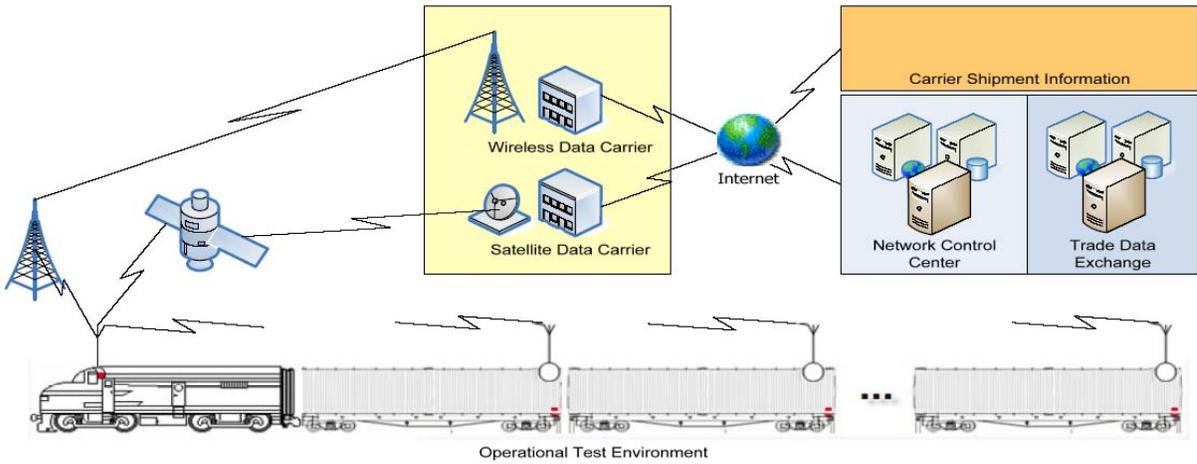


Figure 1: System overview.

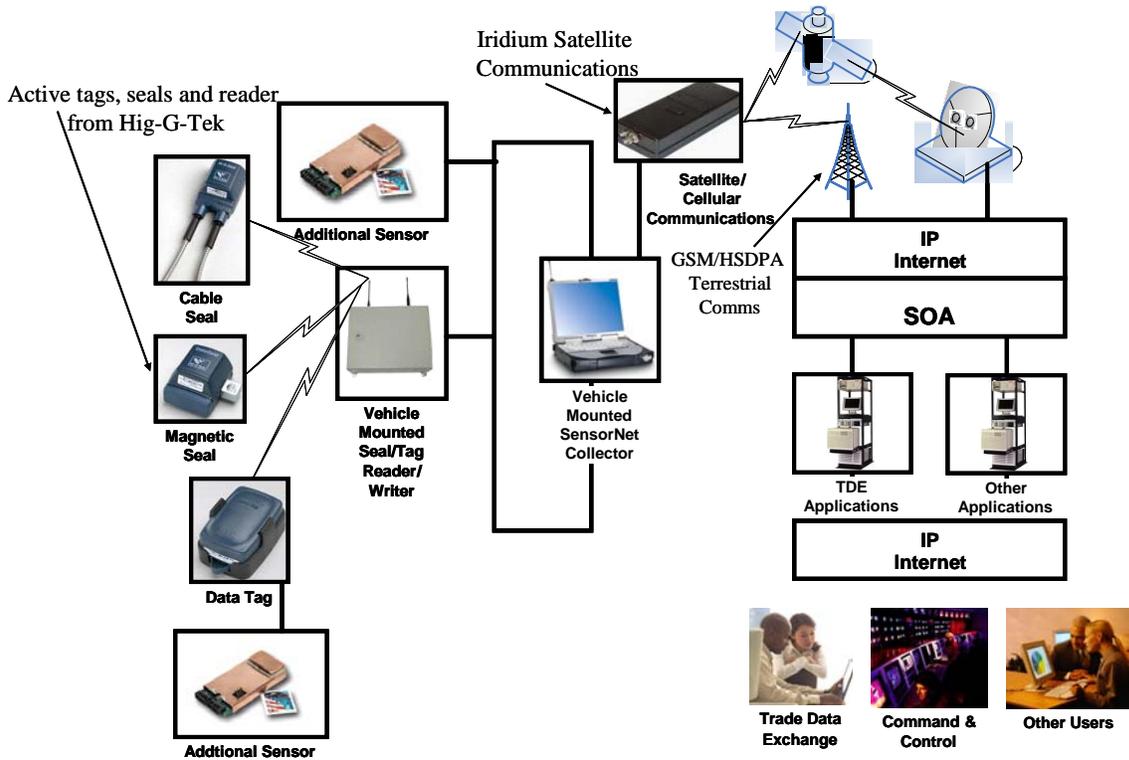


Figure 2: Selected technologies used for proof-of-concept.



Figure 3: Image of the short haul test. The small box attached to the top-front of the locomotive contains the external electronics box (described below).



Figure 4: Electronics box outside of locomotive during long haul test.



Figure 5: Placement of sensor on container.



Figure 6: Train passing through central Mexico during long haul test.



Figure 7: Partial route of long haul test northbound from San Luis Potosi.

```

NOC_AlarmReportingService:
  Date-Time: 2009.07.30 10:33:48 CDT / 2009.07.30 15:33:48 UTC
  Lat/Lon: 25.12046/-101.10943, Quality: Good
  http://maps.google.com/maps?q=25.12046,-101.10943
  TrainId=LngHaulMx
  Severity: Information
  Type: SensorLimitReached
  Message: SensorType=Seal SensorID=IAHA01054318 Event=Open Msg=
  NOC Host: laredo.ittc.ku.edu
  
```

```

Shipment Data:
  Car Pos: 4
  Equipment Id: EDS 53403
  BIC Code: ITTC746485
  STCC: 2643137
  
```

Figure 8: Sample report sent by the TSSN showing integration of TDE and sensor information.

The trials demonstrated a successful technology integration and achieved all of the major goals, as well as validating the overall architecture. Sensors were used to sense container safety on a moving train, reliably communicated by satellite to a virtual network operations center (VNOC), the VNOC demonstrated secure communications with the TDE and fuse sensor and manifest data to deliver near-real-time alerts to decision-makers.

3.2. Task 2: System architecture development, design, measurement and modeling for the SmartPort intelligent transportation systems, with a focus on identification of bottlenecks and scaling issues

In this section, we present a description of this task. For clarity, we present separate sections for the architecture, and for modeling and evaluation.

3.2.1. Transportation Security SensorNet Architecture

The system architecture was designed after careful consideration of existing technologies and architectures [3, 4, 6, 8, 9]. The developed architecture is described in detail [13]. They are briefly summarized here.

Figure 8 gives a high-level overview of the TSSN architecture. The sensors are deployed within a Mobile Rail Network (MRN), which directly communicates with the container sensors and through back-haul communications to the Virtual Network Operations Center (VNOC). The primary task of the MRN is to monitor the sensors on the train and securely communicate important events to the VNOC. The prototype MRN has the capability of using cellular networks when available, and satellite communications otherwise, and thus can be used anywhere in the world. The MRN is discussed in more detail below. The Virtual Network Operations Center (VNOC) is responsible for communicating with the MRN, collecting manifest and other information from the Trade Data Exchange (TDE), processing the information, and sending alarms to the appropriate stakeholder.

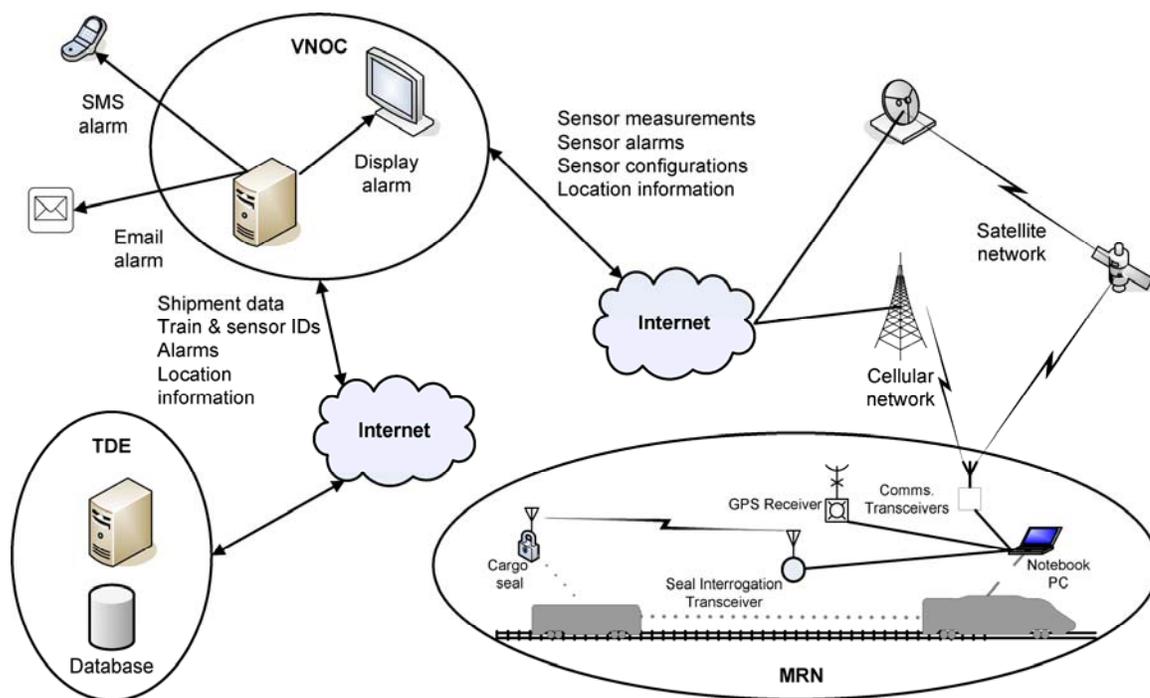


Figure 9: High-level architecture of the TSSN.

The architecture of the MRN is further refined in Figure 10. Here, we can see that the MRN includes a considerable amount of decision-making capability. The TSSN architecture is developed nearly completely around the Services Oriented Architecture (SOA), and Web Services in particular, and thus conforms to open standard interfaces.

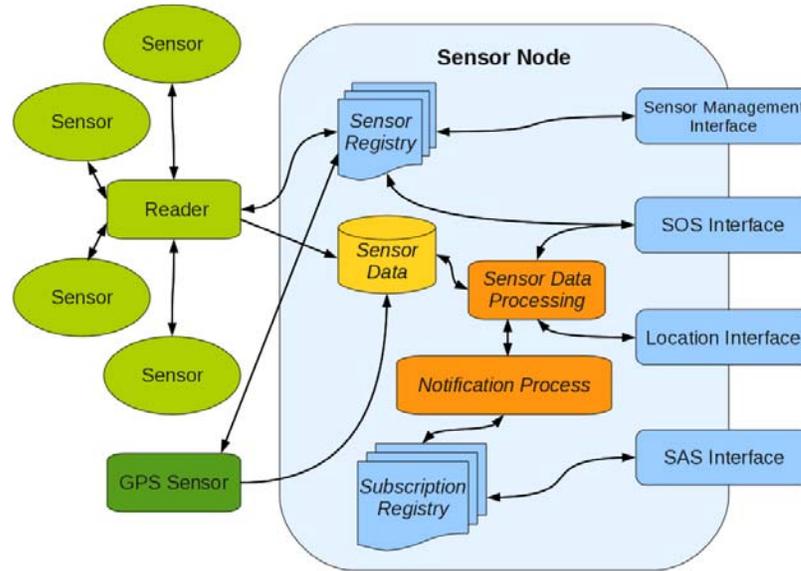


Figure 10: Architecture of the MRN Sensor Node.

The VNOC is expanded in Figure 11. The VNOC is the primary “hub” of communications within the TSSN, and contains most of the decision-making intelligence. The VNOC is comprised of a number of services and interfaces. Essentially, the VNOC receives events from the mobile rail network (MRN), processes those events, obtains manifest and other information from the TDE if necessary, and sends alerts to decision-makers. The VNOC uses rules, historical data, and complex event processing to make real-time decisions. The goal is for the decision-maker to get relevant data and only relevant data in a timely way. The system evaluation is given in more detail in the following section.

3.2.2. Transportation Security SensorNet Measurement and Modeling

An additional goal of the system was to better understand the performance characteristics of the deployed system, to determine how to scale the technology, to evaluate alternative systems and optimize the system design. The first goal was met through extensive system instrumentation primarily through data logging. The second two goals were met using analytical models. We begin this section with an overview of the measured results of the TSSN.

To support measurement, data was collected over all the truck trials, and both short- and long-haul trail. Detailed measurements were taken during both trials, and are reported in detail in [11, 21]. Through private communication, we learned that rail carriers desire a response to a tamper event within 15 minutes, which was taken to be a critical deadline.

To assess performance, consider the sequence of events shown in Figure 12. This shows the typical sequence of messages that flow between services within the TSSN, grouped into epochs. For all trials, including preliminary truck trials and the two rail trials, we kept detailed logs of events, which are summarized in Table I. Considering worst-case times for each epoch, only Epoch 5 took a significant amount of time (less than one minute). The source of that delay was sending an SMS message over a provider network. While the number of observations was not sufficient to make an

accurate estimate of the probability of meeting the 15 minute requirement, the data suggests that the probability is very high.

Based on our experience, we have found a critical system bottleneck. The chosen sensors have limited range when attached to containers. Through testing, we found that we could only monitor about five rail cars reliably, though it was possible to unreliably monitor cars seven to eight cars away. Also, the sensors have essentially no peer-to-peer capability. This is unsuitable for complete rail coverage on long trains. While more capable networking systems are available, such as 802.15.4 and Zigbee, those systems are likely not sufficiently power-efficient nor do they scale sufficiently well in transportation environments. Thus, we believe that there is a need to develop wireless sensor systems capable of peer-to-peer networking using lightweight routing protocols. We are currently collaborating with partners to investigate ways to do that.

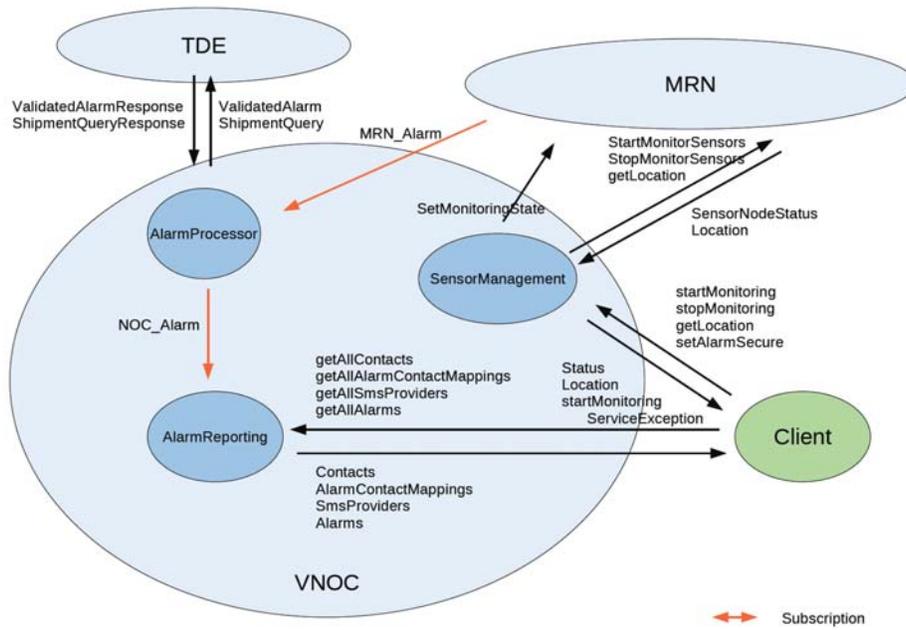
While measurement was important and insightful to gain a better understanding of the system on a detailed level, it leaves open critical system-level questions. For example, would a rack-side communication system work better than train-based? To answer this question, we constructed analytical models, which allowed us to model and evaluate various system-level tradeoffs and optimize the system. The models were based on integer linear programming. The critical result of these models is that under realistic conditions, the models strongly indicate that sensor networks such as the TSSN significantly lower transportation costs. The details of this work is reported in [25] and summarized below.

First, we say that a container is *visible* if a sensor is able to correctly report a critical sensor event within sufficient time. Thus, if a container is visible, then a response team will be able to arrest or deter a container breach. We considered a number of specific system trade-offs, and for clarity we present a few representative experimental results.

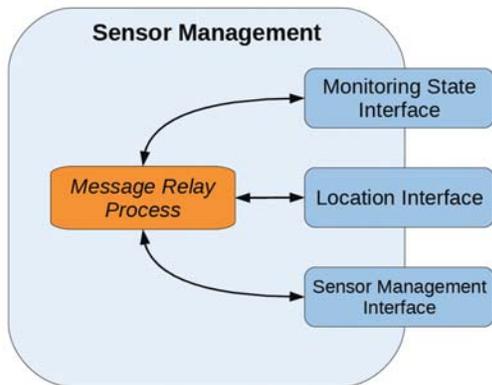
First, we consider optimal sensor placement on a train. The objective was maximum visibility, that is, events are detected with a certain probability and reported in a timely manner, and the probability of a false alarm is kept below a specified threshold. Because the sensors have a limited communication distance, we assume that every third car has a repeater. We assume 66 containers are placed on a 30-car train (due to memory constraints of some of the algorithms and tools used). Here, we assume we are constrained by having only 12 sensors to place on containers. Under these and other conditions detailed in [25], we were able to find an optimal sensor placement. The optimal arrangement of sensors is plotted in Figure 13a.

Next, consider the same scenario except that we now vary the number of sensors available and compute the expected system cost. Here, *cost* includes all costs, including the cost of sensors, communication system, the cost of theft, false alarms, etc. (Again, the details are presented in [25].) That result is shown in Figure 13b. That result clearly shows that costs reduce when visibility increases, i.e., as the number of containers that have sensors increases. While these models include cost and probability estimates, since in many cases actual costs and probabilities are not known, the results show that under assumptions we believe are reasonable, there is significant economic value to using sensor networks to secure cargo.

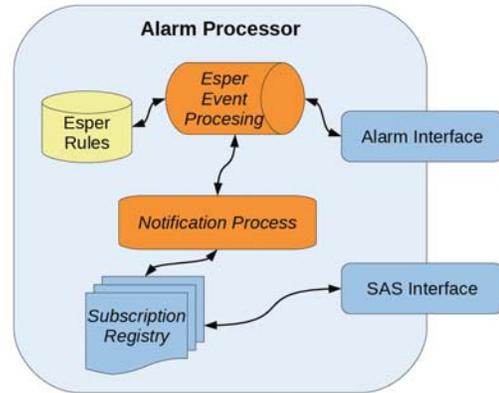
Next, we examined how track-side communications affected the price and performance of the system. Obviously, with track-side communication, the spacing between track-side readers is directly proportional to the expected delay between an event and the ability to report that event. As the expected time to report the event, or the *expected reporting deadline*, increases, the number of track-side readers decrease, and thus the system costs also decrease, which is verified by the results shown in Figure 14a. A trackside system is also highly sensitive to the cost of the reader, which was explored and evaluated, and some results are presented in Figure 14b.



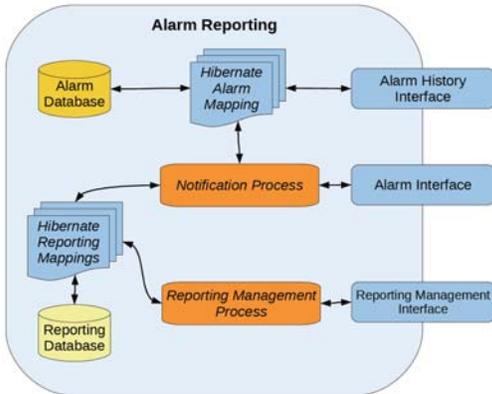
(a) VNOc message overview.



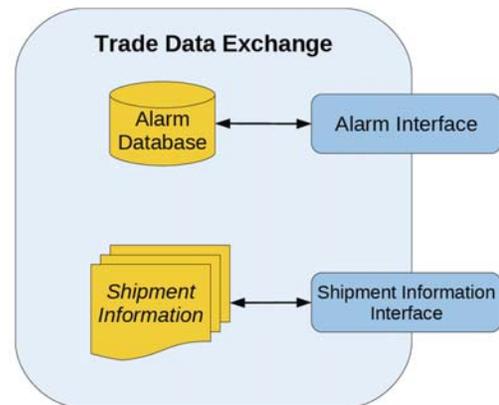
(b) VNOc Sensor Management service.



(c) VNOc Alarm Processor service



(d) VNOc Alarm Reporting service.



(e) TDE interface from VNOc

Figure 11: Architecture of the Virtual Network Operations Center (VNOc).

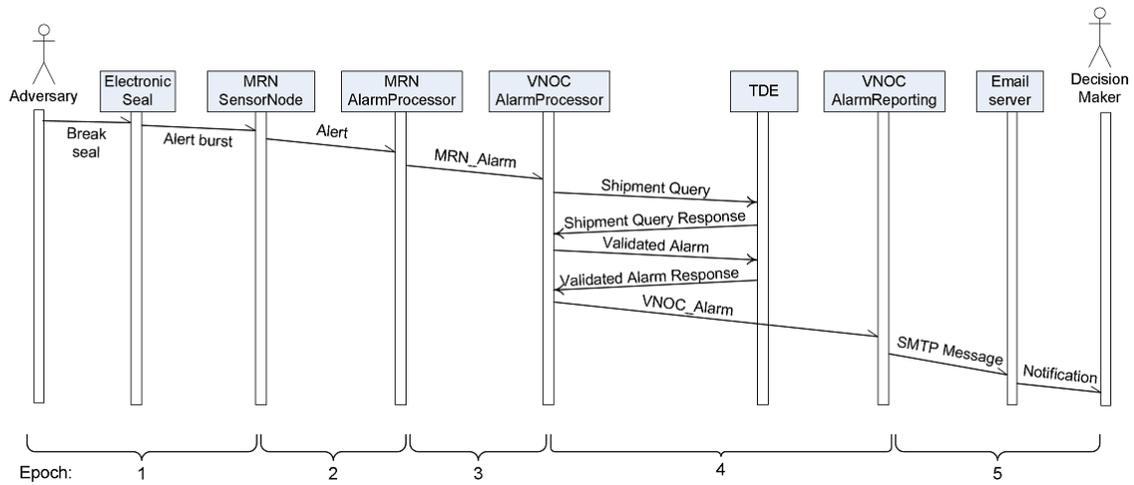


Figure 12: Typical sequence of events that take place from a tamper event to notification of a decision-maker.

Table I: Summary of time statistics for decision-maker notification.

Epoch	Description	Min (s)	Mean (s)	Max (s)
1	Event occurrence to Alert generation	0.81	2.70	8.75
2	Alert generation to MRN AlarmProcessor service	0.01	0.02	0.08
3	One-way delay from MRN AlarmProcessor to VNOc AlarmProcessor	0.45	1.89	2.90
4	MRN Alarm arrival at VNOc AlarmProcessor to AlarmReporting service	0.01	0.17	3.01
5	Elapsed time from VNOc AlarmReporting service to mobile phone	5.2	11.9	58.7

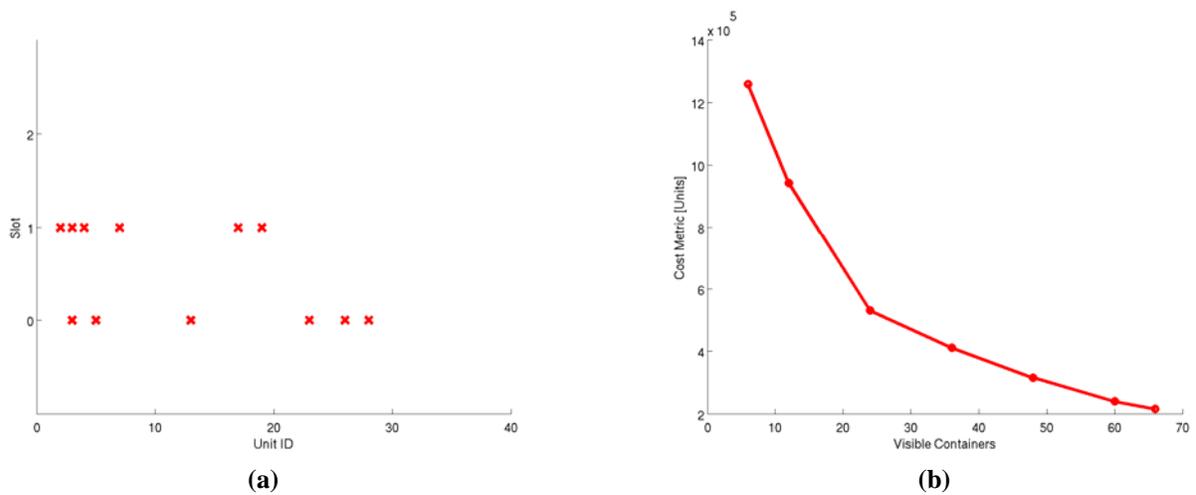


Figure 13: Optimal sensor locations (a), where Unit ID 0 is the locomotive and slot represents elevation of double-stacked containers, and visibility vs. cost (b) obtained by varying the number of sensors.

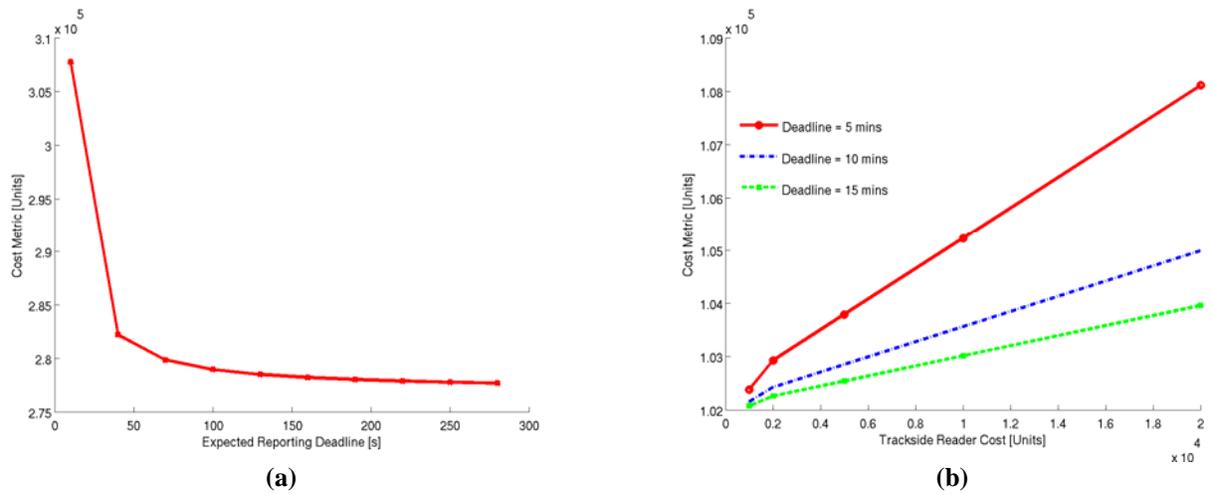


Figure 14: Cost of track-side system varying the required response time (a) and both the trackside reader cost and response deadline (b).

Finally, we compare the economics of a train-mounted communication system vs. a track-side communication system. The results in Figure 15 clearly show the advantage of the track-side reader system. This is because a single reader is able to serve on average 14 trains per week, while a train-mounted system serves only one train. Here, we assume a track-side reader costs approximately 1,000 units (with a unit as roughly equivalent to one US dollar). Obviously, if track-side readers are substantially more expensive, then the outcome would be different.

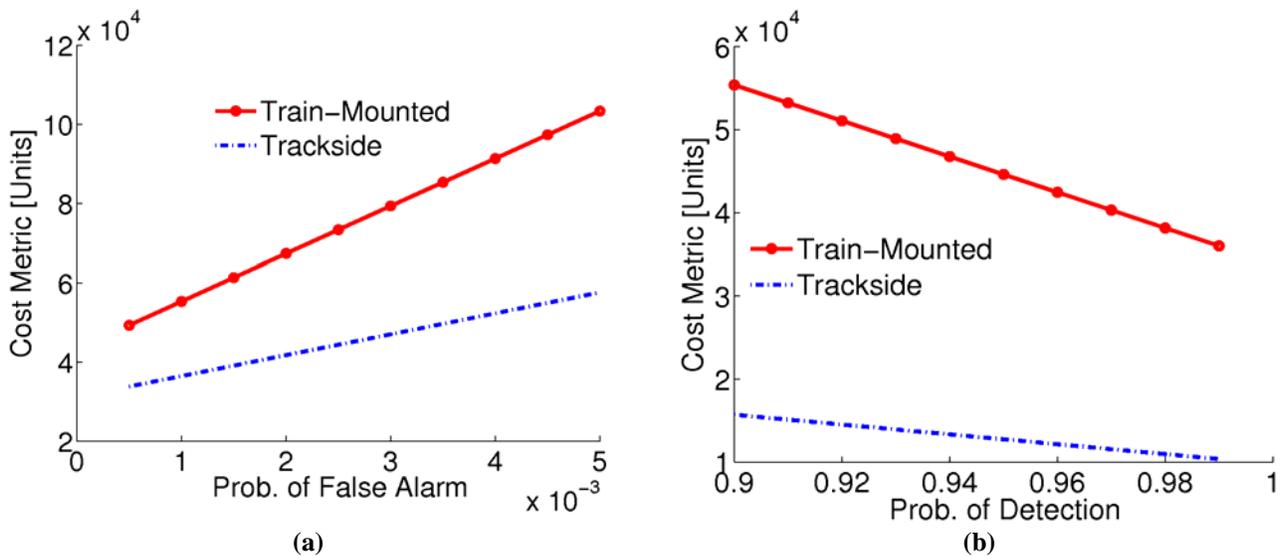


Figure 15: Comparison of train-mounted and trackside communication systems.

From these data, we conclude that we have a robust cost model that can be used to evaluate numerous system tradeoffs. Based on current estimates, a track-side communications system may be more cost-effective than a train-mounted communications system. However, based on our estimated

costs and probabilities (detailed in [25]), it is *always* economically advantageous to use sensor networks to monitor cargo shipments.

3.3. Task 3: Data integration and processing, e.g., controlling the storing and access of information in the SmartPort data clearinghouse

The trusted corridor concept relies on a clearinghouse that integrates information from both the sensors via a Web services interface and related databases, e.g., external logistical and intelligent transportation systems such as the TDE. Secure communication between the TSSN and TDE has been designed and implemented using well-established standards. Thus, the architecture is standards-based and open. The overall system architecture is described in [13]. An overview of the security aspects is given in Figure 16. The TDE–TSSN security relies on three layers: 1) firewalls to let traffic in only to and from certain computers and port numbers, 2) HTTPS (secure sockets) encryption layer for privacy, and 3) the use of username and password tokens for authentication. This approach was taken given practical considerations of interfacing with legacy software, and provides adequate security. In a related project, this feature has been tested with multiple TSSNs communicating with the TDE, illustrating the security and scalability of the system.

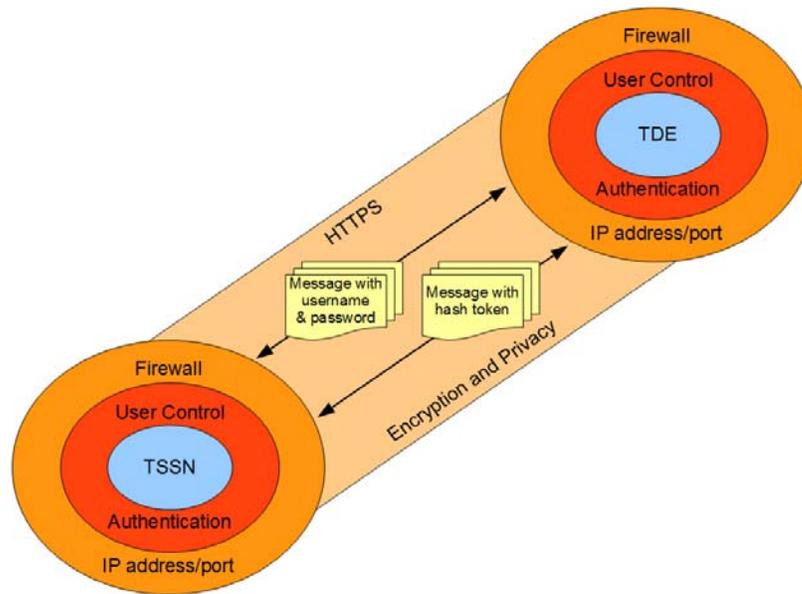


Figure 16: Security architecture used between TSSN and TDE.

One of the notable outcomes of this effort was that we found a possible architectural flaw with the Web Services Architecture (WSA). The conflict arises when one uses the Publish / Subscribe paradigm, a preferred method for efficient data exchange (as opposed to polling) and publishing the policy of a security service. The details are found in [19], but essentially there is no mechanism for discovering the security policy of a service when using the publish / subscribe paradigm. What can be done is “hard code” the security policy within the two communicating services, which was done for this project, but that is obviously not ideal, and contrary to the spirit of the Web Services architecture. We believe that this is a significant finding and an architectural flaw that needs to be addressed by the Web Services community.

3.4. Task 4: Communications technologies to enable continuous monitoring

One of the unique challenges for communication in this environment is that most web services implementations assume connection-oriented services. Due to economics and other practical limitations, we can not assure continuous connection to rail, especially in parts of the world with no wireless coverage and in mountainous regions where satellite communications can be sporadic. We extensively surveyed technologies that would be useful for continuous monitoring of cargo on rail. The results are summarized in [3, 8, 9].

We determined the need to develop a messaging service that would seamlessly integrate into the Web Services paradigm. This is documented in detail in [22], and described briefly below. The affected clients and servers were modified to communicate through asynchronous messaging services, which is possible through the Axis2 implementations using callbacks. Then, the transport mechanism (normally HTTP or HTTPS) was replaced with a Java Messaging Service (JMS), using the ActiveMQ implementation of JMS. JMS is an asynchronous message passing service that is robust under sporadic connections. When JMS detects a connection, it exchanges messages through a shared queue paradigm. Figure 17 illustrates the distributed queuing model used to exchange messages.

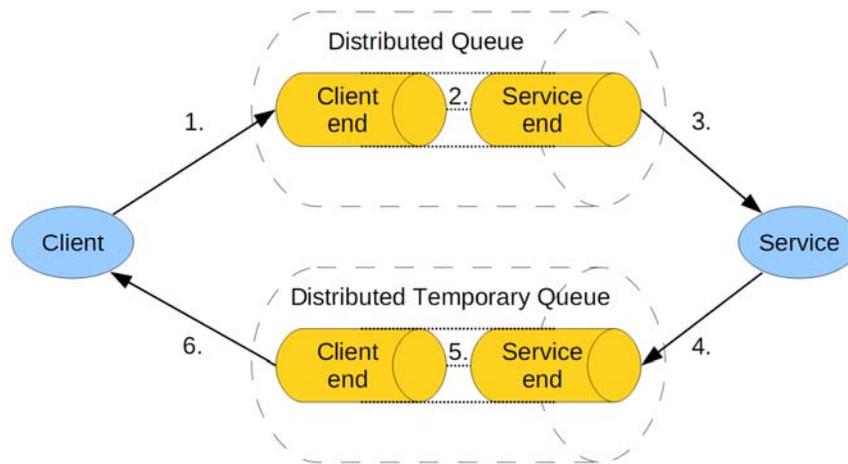


Figure 17: Distributed shared queue model of message exchanges for unreliable communications channel.

During the long haul test in Mexico, we recorded data from the JMS service. The results are shown in Figure 18, where the dwell time of each message in the queue is plotted vs. time. At the bottom, a line shows when the satellite communications channel was connected. As the figure clearly shows, messages were usually exchanged rather quickly when the communications channel was established, and buffered, sometimes for considerable time, when the channel was down or busy.

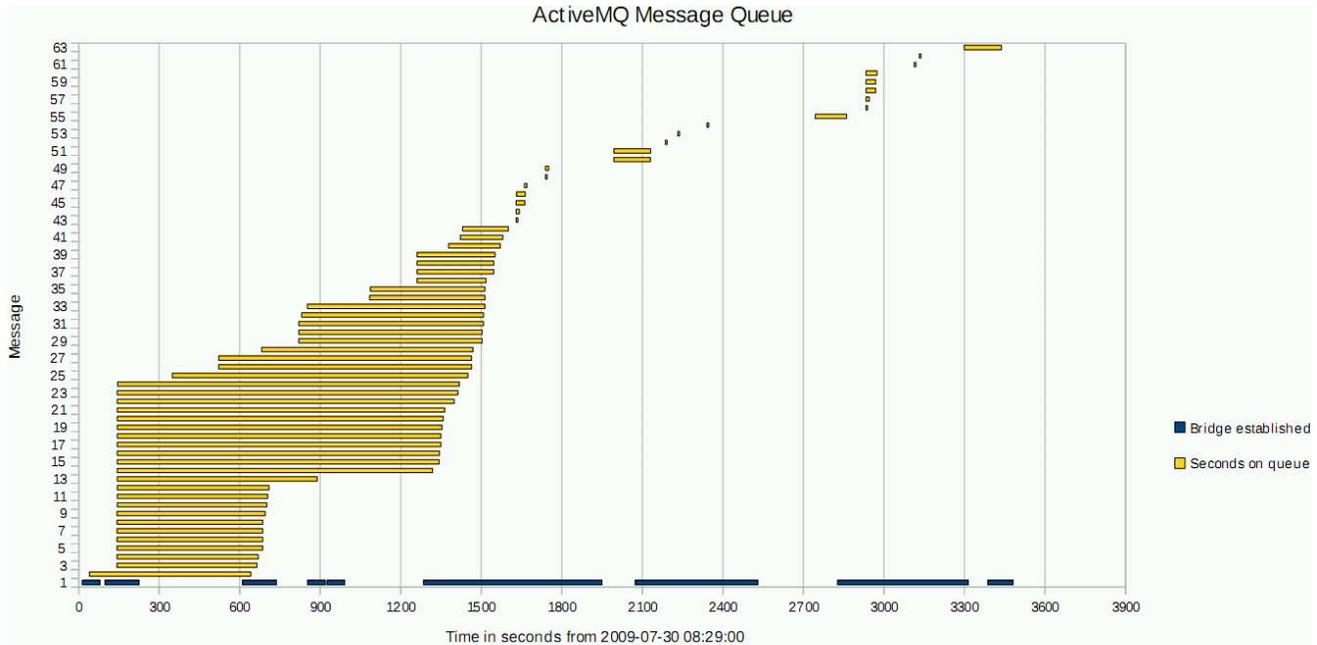


Figure 18: Measured results of message dwell times during the long haul trial.

3.5. Task 5: RFID applications

We studied the use of RFID technologies for cargo applications. This resulted in the development of a long-range passive UHF RFID tag that was capable of identification at long distances [23], and a survey of commercially-available location-positioning technologies [14]: RF Controls [26] and Mojix [27]. We summarize our findings below.

- As claimed, Mojix system is extremely sensitive. It was able to detect the developed long-range passive UHF RFID tag at nearly 130 feet away when the Star was pointing away from the tag (an adverse condition).
- We observed location-positioning capabilities that were not highly refined and took some time to settle on an accurate location. Regardless, it showed location positioning accuracy sufficient to one to two meters, which is sufficient for an intermodal facility.
- Systems continue to improve since the writing of [23], including the use of wireless nodes in the Mojix system. A mobile eNode could be used to inventory and determine location of containers within the yard periodically.
- The developed RFID tag provided excellent long-distance performance. Improvement in the integrated circuit (IC) has been growing steadily so that read distances double every two to three years. While current IC design is meeting fundamental limits in passive design, low-cost battery technology continues to develop and may provide continued performance gains.

We conclude that the use of passive RFID in intermodal facilities is possible given existing technology, but would require extensive deployment of infrastructure and a community-wide commitment.

4. Description of Student Activities

This project used students extensively in the design, development, and testing of the system. The following table shows students who were funded by the project and made contributions to the project as shown by their participation in the following technical reports.

Table II: List of students and their participation on the project.

Student	Title	Reference
Daniel T. Fokum	A Taxonomy of Sensor Network Architectures	[8]
	A Survey on Methods for Broadband Internet Access on Trains	[9]
	Status Update: A Unified Architecture for SensorNet with Multiple Owners: Supplement to Advance SensorNet Technologies to Monitor Trusted Corridors	[10]
	Experiences from a Transportation Security Sensor Network Field Trial	[11]
	Service Oriented Architecture for Monitoring Cargo in Motion Along Trusted Corridors	[13]
	Service Oriented Architecture for Monitoring Cargo in Motion Along Trusted Corridors	[13]
	Summary of Status: A Unified Architecture for SensorNet with Multiple Owners: Supplement to Advance SensorNet Technologies to Monitor Trusted Corridors	[14]
	An Open System Transportation Security Sensor Network: Field Trial Experiences	[21]
Martin Kuehnhausen	Experiences from a Transportation Security Sensor Network Field Trial	[11]
	Service Oriented Architecture for Monitoring Cargo in Motion Along Trusted Corridors	[13]
	Summary of Status: A Unified Architecture for SensorNet with Multiple Owners: Supplement to Advance SensorNet Technologies to Monitor Trusted Corridors	[14]
	Application of the Java Message Service in Mobile Monitoring Environments	[18]
	Implementing Web Services: Conflicts Between Security Features and Publish/Subscribe Communication Protocols	[19]
	Framework for Analyzing SOAP Messages in Web Service Environments	[20]
	An Open System Transportation Security Sensor Network: Field Trial Experiences	[21]
Pradeepkumar Mani	A Taxonomy of Sensor Network Architectures	[8]
Satyasree Muralidharan	A Taxonomy of Sensor Network Architectures	[8]
Angela N. Oguna	Experiences from a Transportation Security Sensor Network Field Trial	[11]

	Service Oriented Architecture for Monitoring Cargo in Motion Along Trusted Corridors	[13]
	Summary of Status: A Unified Architecture for SensorNet with Multiple Owners: Supplement to Advance SensorNet Technologies to Monitor Trusted Corridors	[14]
	Transportation security Sensor Network: Sensor Selection and Signal Strength Analysis	[17]
	An Open System Transportation Security Sensor Network: Field Trial Experiences	[21]
Matthew Zeets	Experiences from a Transportation Security Sensor Network Field Trial	[11]
	Service Oriented Architecture for Monitoring Cargo in Motion Along Trusted Corridors	[13]
	Summary of Status: A Unified Architecture for SensorNet with Multiple Owners: Supplement to Advance SensorNet Technologies to Monitor Trusted Corridors	[14]
	An Open System Transportation Security Sensor Network: Field Trial Experiences	[21]

5. Conclusions

The goal of the project was to develop and integrate systems that provide the ability to track and monitor the security of cargo in transit. To achieve the goal, we developed the Transportation Security SensorNet (TSSN) to monitor rail cargo with the SmartPort Trade Data Exchange (TDE) system. The system was demonstrated in both a “short haul” and “long haul” (international) shipping scenario. Data from the tests were taken and extensively analyzed, identifying bottlenecks and new research directions, including the use of security and notification events in Web Services, and the need for efficient, long-range wireless sensors appropriate for the rail environment.

SmartPort continues to develop and commercialize the TDE, and has funded additional activity to further integrate TSSN and the TDE to support the exchange of custody through sensors and mobile communication systems, as well as continue to look for opportunities to collaborate on developing the necessary technologies to enable full coverage of a train, container yard, and other intermodal facilities and transports.

6. References

- [1] Pradeepkumar Mani, David W. Petr, “Clique Number vs. Chromatic Number in Wireless Interference Graphs: Simulation Results,” University of Kansas, Lawrence, KS, Tech. Rep. ITTC-FY2007-TR41420-01, October 2007.
- [3] S Muralidharan, V. S. Frost, and G. J. Minden , “A Framework for Sensor Networks with Multiple Owners,” University of Kansas, Lawrence, KS, Tech. Rep. ITTC-FY2008-TR-41420-03; December 2007.
- [4] Daniel T. Fokum, Dr. Victor S. Frost, and Dr. Gary J. Minden, “An Evaluation of Sensing Platforms Used for Sensor Network Research,” University of Kansas, Lawrence, KS, Tech. Rep. ITTC-FY2008-TR-41420-04, December 2007.
- [5] Andrew Boie and Douglas Niehaus, “Performance Constraints of Distributed Control Loops on Linux Systems,” University of Kansas, Lawrence, KS, Tech. Rep. ITTC-FY2008-TR-41420-05, December 2007.
- [6] Gary Minden, Victor Frost, David Petr, Douglas Niehaus, Ed Komp, Daniel Fokum, Pradeepkumar Mani, Andrew Boie, Satyasree Muralidharan, and James Stevens, “Phase One Report: A Unified Architecture for SensorNet with Multiple Owners,” University of Kansas, Lawrence, KS, Tech. Rep. ITTC-FY2008-TR-41420-06; December 2007.
- [7] Leon S. Searl, “Service Oriented Architecture for Sensor Networks Based on the Ambient Computing Environment,” University of Kansas, Lawrence, KS, Tech. Rep. ITTC-FY2008-TR-41420-07; February 2008.
- [8] D.T. Fokum, V.S. Frost, P. Mani, G.J. Minden, J.B. Evans, and S. Muralidharan, “A Taxonomy of Sensor Network Architectures,” University of Kansas, Lawrence, KS, Tech. Rep. ITTC-FY-2009-TR-41420-08, July 2008.
- [9] Daniel T. Fokum and Victor S. Frost, “A Survey on Methods for Broadband Internet Access on Trains,” University of Kansas, Lawrence, KS, Tech. Rep. ITTC-FY-2009-TR-41420-09, August 2008.
- [10] Victor S. Frost, Gary J. Minden, Joseph B. Evans, Daniel T. Fokum, “Status Update: A Unified Architecture for SensorNet with Multiple Owners: Supplement to Advance SensorNet Technologies to Monitor Trusted Corridors,” University of Kansas, Lawrence, KS, Tech. Rep. ITTC-FY2009-TR-41420-10, August, 2008.
- [11] Daniel T. Fokum, Victor S. Frost, Daniel DePardo, Martin Kuehnhausen, Angela N. Oguna, Leon S. Searl, Edward Komp, Matthew Zeets, Joseph B. Evans, Gary J. Minden, “Experiences from a Transportation Security Sensor Network Field Trial,” University of Kansas, Lawrence, KS, Tech. Rep. ITTC-FY2009-TR-41420-11, June 2009.

- [12] Victor S. Frost, Gary J. Minden, and Joseph B Evans, "Summary of Status: A Unified Architecture for SensorNet with Multiple Owners: Supplement to Advance SensorNet Technologies to Monitor Trusted Corridors," University of Kansas, Lawrence, KS, Tech. Rep. ITTC-FY-2009-TR-41420-12, December 2008.
- [13] Martin Kuehnhausen, Daniel T. Fokum, Victor S. Frost, Daniel DePardo, Angela N. Oguna, Leon Searl, Edward Komp, Matthew Zeets, Daniel D. Deavours, Joseph B. Evans, and Gary J. Minden, "Service Oriented Architecture for Monitoring Cargo in Motion Along Trusted Corridors," University of Kansas, Lawrence, KS, Tech. Rep. ITTC-FY2010-TR-41420-13, July 2009.
- [14] Daniel D. Deavours, "Application of Passive UHF RFID in Intermodal Facilities," University of Kansas, Lawrence, KS, Tech. Rep. ITTC-FY2010-TR-41420-14; July 2009.
- [15] V.S. Frost, G.J. Minden, J.B. Evans, L. Searl, D.T. Fokum, D. Deavours, E. Komp, A. Oguna M. Zeets, M. Kuehnhausen, D. Depardo, "Summary of Status: A Unified Architecture for SensorNet with Multiple Owners: Supplement to Advance SensorNet Technologies to Monitor Trusted Corridors," University of Kansas, Lawrence, KS, Tech. Rep. ITTC-FY2010-TR-41420-15; July 2009.
- [16] EDS, an HP Company, J. Walther, L. Sackman, M. Gatewood, J. Spector, S. Hill, J. Strand, "EDS HP Final Report," University of Kansas, Lawrence, KS, Tech. Rep. ITTC-FY2010-TR-41420-16; December 2009.
- [17] Angela Oguna, "Transportation Security Sensor Network: Sensor Selection and Signal Strength Analysis," University of Kansas, Lawrence, KS, Tech. Rep. ITTC-FY2010-TR-41420-17; December 2009.
- [18] M. Kuehnhausen and V. S. Frost, "Application of the Java Message Service in Mobile Monitoring Environments," University of Kansas, Lawrence, KS, Tech. Rep. ITTC-FY2010-TR-41420-18; December 2009.
- [19] E. Komp, V. Frost, and M. Kuehnhausen, "Implementing Web Services: Conflicts Between Security Features and Publish/Subscribe Communication Protocols," University of Kansas, Lawrence, KS, Tech. Rep. ITTC-FY2010-TR-41420-19; February 2010.
- [20] M. Kuehnhausen and V. S. Frost, "Framework for Analyzing SOAP Messages in Web Service Environments," University of Kansas, Lawrence, KS, Tech. Rep. ITTC-FY2010-TR-41420-20, March, 2010.
- [21] D. T. Fokum, V. S. Frost, D. DePardo, M. Kuehnhausen, A. N. Oguna, L. S. Searl, E. Komp, M. Zeets, D. D. Deavours, J. B. Evans, and G. J. Minden, "An Open System Transportation Security Sensor Network: Field Trial Experiences," University of Kansas, Lawrence, KS, Tech. Rep. ITTC-FY2010-TR-41420-21; March 2010.

- [22] M. Kuehnhausen and V. S. Frost, "Transportation Security SensorNet: A Service Oriented Architecture for Cargo Monitoring," University of Kansas, Lawrence, KS, Tech. Rep. ITTC-FY2010-TR-41420-22; April, 2010.
- [23] S. Aroor and D. D. Deavours, "A Dual-Resonant Microstrip-Based UHF RFID 'Cargo' Tag," University of Kansas, Lawrence, KS, Tech. Rep. ITTC-FY2010-TR-41420-23; March, 2008.
- [24] Pradeepkumar Mani, Satyasree Muralidharan, Victor S. Frost, Gary J. Minden, and David W. Petr, "Unified SensorNet Architecture with Multiple Owners: An Implementation Report," University of Kansas, Lawrence, KS, Tech. Rep. ITTC-FY2010-TR-41420-24; May, 2010.
- [25] Daniel T. Fokum and Victor S. Frost, "Modeling for Analysis and Design of Communications Systems and Networks for Monitoring Cargo in Motion Along Trusted Corridors," University of Kansas, Lawrence, KS, Tech. Rep. ITTC-FY2010-TR-41420-25; May, 2010.
- [26] RF Controls, LLC., "Welcome to RF Controls!", available <http://www.rfctrls.com/index.html>, accessed June 16, 2010.
- [27] "Mojix Star System Brochure," Mojix, 2008, available http://www.mojix.com/products/documents/Mojix_STAR_System.pdf, accessed June 16, 2010.

List of Appendices

The following is a list of the appendices for this report. Each appendix is also a technical report.

Appendix Number	ITTC TR #	Title
A.1	ITTC-FY2008-TR41420-07	Service Oriented Architecture for Sensor Networks Based on the Ambient Computing Environment
A.2	ITTC-FY2009-TR41420-08	A Taxonomy of Sensor Network Architectures
A.3	ITTC-FY2009-TR41420-09	A Survey on Methods for Broadband Internet Access on Trains Status Update: A Unified Architecture for SensorNet with Multiple Owners: Supplement to Advance SensorNet Technologies to Monitor Trusted Corridors
A.4	ITTC-FY2009-TR41420-10	Experiences from a Transportation Security Sensor Network Field Trial Summary of Status: A Unified Architecture for SensorNet with Multiple Owners: Supplement to Advance SensorNet Technologies to Monitor Trusted Corridors
A.5	ITTC-FY2009-TR41420-11	Experiences from a Transportation Security Sensor Network Field Trial Summary of Status: A Unified Architecture for SensorNet with Multiple Owners: Supplement to Advance SensorNet Technologies to Monitor Trusted Corridors
A.6	ITTC-FY2009-TR41420-12	Experiences from a Transportation Security Sensor Network Field Trial Summary of Status: A Unified Architecture for SensorNet with Multiple Owners: Supplement to Advance SensorNet Technologies to Monitor Trusted Corridors
A.7	ITTC-FY2010-TR41420-13	Service Oriented Architecture for Monitoring Cargo in Motion Along Trusted Corridors
A.8	ITTC-FY2010-TR41420-14	Application of Passive UHF RFID in Intermodal Facilities Summary of Status: A Unified Architecture for SensorNet with Multiple Owners: Supplement to Advance SensorNet Technologies to Monitor Trusted Corridors
A.9	ITTC-FY2010-TR41420-15	Application of Passive UHF RFID in Intermodal Facilities Summary of Status: A Unified Architecture for SensorNet with Multiple Owners: Supplement to Advance SensorNet Technologies to Monitor Trusted Corridors
A.10	ITTC-FY2010-TR41420-16	EDS HP Final Report
A.11	ITTC-FY2010-TR41420-17	Transportation Security Sensor Network: Sensor Selection and Signal Strength Analysis
A.12	ITTC-FY2010-TR41420-18	Application of the Java Message Service in Mobile Monitoring Environments
A.13	ITTC-FY2010-TR41420-19	Implementing Web Services: Conflicts Between Security Features and Publish/Subscribe Communication Protocols
A.14	ITTC-FY2010-TR41420-20	Framework for Analyzing SOAP Messages in Web Service Environments
A.15	ITTC-FY2010-TR41420-21	An Open System Transportation Security Sensor Network: Field Trial Experiences
A.16	ITTC-FY2010-TR41420-22	Transportation Security SensorNet: A Service Oriented Architecture for Cargo Monitoring
A.17	ITTC-FY2010-TR41420-23	A Dual-Resonant Microstrip-Based UHF RFID ‘Cargo’ Tag
A.18	ITTC-FY2010-TR41420-24	Unified SensorNet Architecture with Multiple Owners: An Implementation Report
A.19	ITTC-FY2010-TR41420-25	Modeling for Analysis and Design of Communications Systems and Networks for Monitoring Cargo in Motion Along Trusted Corridors



Technical Report

**Service Oriented Architecture for Sensor Networks
Based on the Ambient Computing Environment**

Leon S. Searl

ITTC-FY2008-TR-41420-07

February 2008

Project Sponsor:
Oak Ridge National Laboratory

1	Overview.....	1
2	ACE SOA Infrastructure.....	1
2.1	Client/Service Communication.....	3
2.1.1	Client SOAP Stack.....	4
2.1.2	Service SOAP Stack	5
2.1.3	Event Notification.....	5
2.2	Service Directory Service	6
2.2.1	Service Publication	7
2.2.1.1	ASD Service Alive Status.....	8
2.2.2	Service Discovery by Client	8
2.3	Authentication Service.....	9
2.4	Authorization Service	9
2.5	Remote Services.....	9
3	Client and Service Development	10
4	Scenario Examples.....	10
4.1	Client/Service Messaging Example plus Remote Service Example.....	10
4.2	ACESOA Federation Example	24
4.3	Event Notification Examples	35
4.3.1	Service – Service Notification	35
4.3.2	Client - Service – Service Notification	36
4.3.3	Client – Service Notification	37

1 Overview

ACE SOA (Ambient Computing Environment)(Service Oriented Architecture) is the forth generation ACE implementation for providing distributed computing, media and sensing services to service consumers (clients) in a dispersed environment. ACE SOA is the infrastructure providing message and data communication, confidentiality, authenticity and permissions plus service discovery within an enterprise and between enterprises. ACE also provides a framework for developing new services and clients of services.

ACE SOA is a reimplementation of the original ideas of ACE but utilizing current technology and widely accepted Web Service specifications and publicly available implementations which are suitable for Sensor Networks. Some of the Web Service specifications in use are SOAP, the WS-X specifications and UDDIv3.

2 ACE SOA Infrastructure

ACE SOA is an infrastructure to allow Web Service based clients and services of one or more enterprises to interact using the following features:

- Provide means for service to publish its URL location and Web Service Interface for discovery by clients.
- Allow clients to discover service's URL location and Web Service Interface.
- Provide a secure communication channel between clients and services.
- Provide mechanism for clients to subscribe to service 'events' or 'alarms'.
- Authenticate a client to a service.
- Provide fine grain authorization of a client's use of a service.
- Provide a framework for development of new clients and services.
- Establish a trust relationship between enterprises.

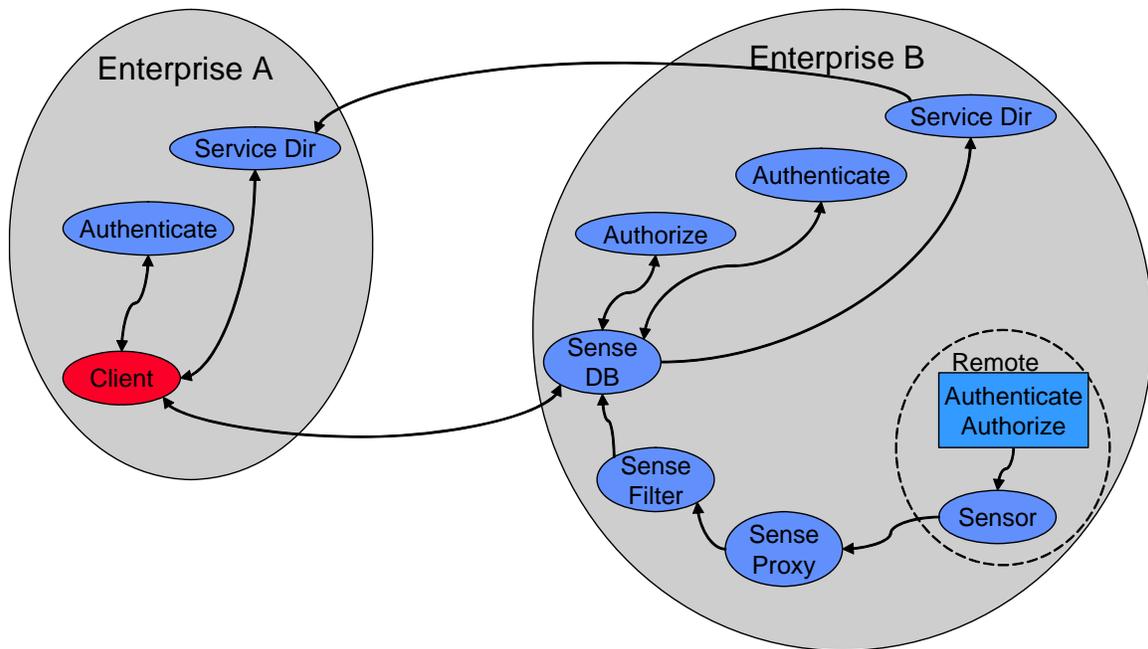


Figure 1 Federation of Services Overview

Figure 1 illustrates a simplified federation of clients and services. The client in this example is in 'red'. The services are in blue. The ACE infrastructure provided services are the 'Service Directory', 'Authenticate' and 'Authorize' services. The remaining services have been developed by the enterprise.

Each federation has its own Service Directory (UDDI), Authentication service and Authorization service (although this example does not show an authorization service for enterprise A).

When a service starts up it publishes its location URL (host and port) and its Web Service interface (operations and operation arguments and argument types) to its Service Directory so that clients are able to find it. In Figure 1 the 'Sense DB' service of Enterprise B has published its information to the 'Service Directory' service.

Service Directories subscribe with each other for new inter-service publications. When a new service that can be used from outside the enterprise publishes with its Service Directory, the Service Directory then notifies any other subscribed Service Directories about the new service. In Figure 1 the Service Directory of Enterprise B has notified the Service Directory of Enterprise A of the service 'Sense DB' from Enterprise B.

A client can discover a service by querying its own Service Directory (query can be by service type or by some other distinguishing category published by a service). The service directory tells the client where the service(s) is and the interface (operations and arguments to the operations) needed to communication with the service. The service may be in the client's own enterprise or it may be in another enterprise. In Figure 1 the Client has queried for a service that is in the 'Sensor Database' category and its Service

Directory has returned the location URL and interface for the ‘Sense DB’ from Enterprise B.

Services may also be clients of other services. In Figure 1 ‘Sense Proxy’ is a client of ‘Sensor’. Service ‘Sense Filter’ is a client of service ‘Sense Proxy’. Service ‘Sense DB’ is a client of ‘Sense Filter’, ‘Authorize’, ‘Authenticate’ and ‘Service Dir’.

The Enterprise A client communicates with the Sense DB service using encrypted messages. The ‘user’ of the client will be authenticated by ‘Sense DB’ using a certificate that the client obtained from its Authenticate service and passed along it’s message.

The service determines if a client/user is authorized to perform the requested operation in the message by querying its Authorize service.

In Figure 1 there is a remote service that is connected by a slow or intermittent connection so instead of communicating with Service Dir, Authenticate and Authorize services it has a ‘Sense Proxy’ service that handles that communication. Only the ‘Sense Proxy’ service is allowed to communicate with the remote service so the remote service can have its authentication and authorization information in local files.

2.1 Client/Service Communication

ACESOA clients communicate with ACESOA services utilizing the open web service standard SOAP (Service Operation Access Protocol) specification. ACESOA only uses version 1.2 of the [SOAP specification](#). The implementation of SOAP used by ACESOA is the [Apache Axis2](#) package.

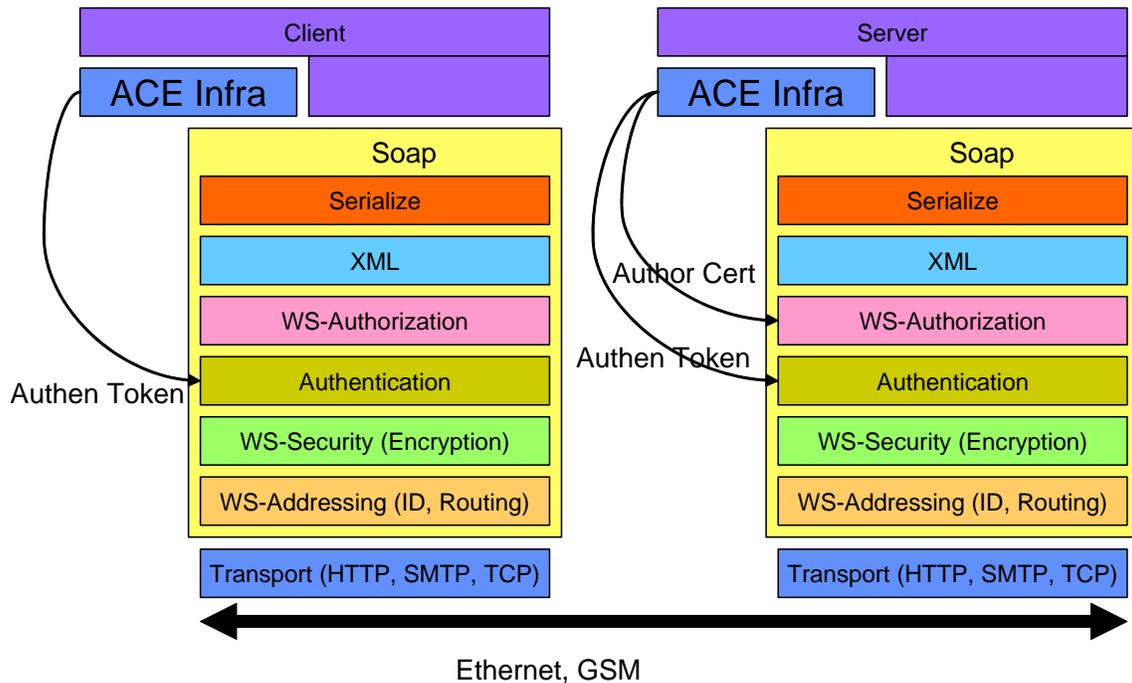


Figure 2 ACESOA Message Stack

Figure 2 shows the SOAP message stack utilized by ACESOA. A message originates with the client application code, passes down through the SOAP layers to the underlying network. The network routes the message to the proper host and port. The message then passes up through the soap layers for the service until it reaches the service application code.

Each client and each service has ACESOA specific code that is used to interface with portions of the SOAP stack (labeled as 'ACE Infra').

The Serialize and XML layers adhere to the SOAP 1.2 specification for the message body. This is integral to Apache Axis2.

The WS-Authorization specification has not been written yet. Until the WS-Authorization is written ACESOA will utilize xACML (extendable Access Control Markup Language) to specify credentials and will make a best guess for the protocol for placing xACML into SOAP messages based a working draft WS-Authorization XML schema found on the web.

The Authentication layer uses the WS-Security mechanism for signing headers and messages. This is implemented by the Apache Rampart module.

The Encryption layer uses the WS-SecureConversation specification. This is implemented by the Apache Rampart module.

WS-Addressing layer follows the like named specification. This is integral to Apache Axis2.

2.1.1 Client SOAP Stack

To invoke a procedure on a service the client must first create the message body. Since ACESOA utilizes the Axis Axiom Object Model the client must create the message as an Axiom Object Model. The Object Model has the procedure name and procedure argument names and argument values placed in it.

The client passes the OM (message object model) to SOAP which then serializes the OM. The serialized object model is placed in XML in the SOAP schema.

On the client side WS-Authorization is skipped.

On the client side the Authentication layer (which is really a part of WS-Security) is given a certificate or token that has been provided by an enterprise authentication Certificate Authority (CA) through the 'Ace Infra' layer. The 'Ace Infra' layer handles this automatically for the client. The token or certificate identifying the user is placed in the message.

The message head and body is encrypted by WS-SecureConversation which is part of WS-Security. WS-Security in Axis2 to is provided by the Rampart module.

SOAP routing information (service being called, session in the service to use and the procedure being called) is placed in the message at the WS-Address layer.

The message is encapsulated and network routing information is added by the transport layer then the message is placed on the network.

2.1.2 Service SOAP Stack

A message from an ACESOA client, destined for an ACESOA service is first received by the transport layer from the network. The SOAP message is extracted from the transport message and delivered to the WS-Addressing layer.

The WS-Addressing layer determines from the message which service the message is for, which session within the service to deliver the message to and which procedure in the service session to invoke.

The header and body of the message are decrypted by WS-SecureConversation in the WS-Security layer. WS-Security is implemented by the Rampart package from Apache.

The authentication of the client user is determined by extracting the token or certificate from the message and verifying the token or certificate with the CA of the service's enterprise. The communication with the CA (Authenticate service) is handled by the 'Ace Infra' layer. To reduce communication overhead with the Authenticate service the token or certificate is cached in the service with a limited cache lifetime. The service also has subscribed to the Authenticate service to be notified if the validity of the user's token or certificate changes. Any failed authentication results in an error message back to the client.

The WS-Authorization layer determines which procedure and the arguments of the procedure that the client is trying to use. To determine if the client is authorized to invoke the procedure with the specified arguments the service's Authorize service is queried. To reduce communication overhead with the Authorize service the user's authorization credential is cached in the service with a limited cache lifetime. The service also has subscribed to the Authorize service to be notified if the validity of the user's credential changes. Any failed authorization results in an error message back to the client.

The message is extracted from the XML and placed in an Axiom Object Model by the XML and serialize layers.

The appropriate method for invoking the procedure is called with the Axiom Object Model as the argument.

2.1.3 Event Notification

An Event is the asynchronous generation of data by a service (an event). Event Notification is the sending of Event data by asynchronous message (publishing) from a service to an event subscriber. An event generating service allows clients to send

subscription messages to it specifying the desired event and the destination for the event notification (publishing).

Event notification in ACE_SOA is based on the WS-Eventing specification and implemented by the Apache Savan module for Apache Axis2.

In Web based SOA, messages are either one way or in-out exchange. An in-out exchange is one message into a service and a single response to the client that originated the message. Clients must always initiate a message exchange. Clients may never receive a one way message nor be the recipient of the first message of an in-out exchange. Because of this Web Service restriction a client can not receive an Event Notification directly.

For a standalone client to receive an Event Notification it is necessary to embed a service server within the client. The service server then starts up a service that can receive Event Notifications. The client must register a callback class/method with the embedded service to be called when the embedded service receives an Event Notification. When the client sends a subscription request to an Event Notice generating service it includes in the request the URI of the embedded service as the delivery endpoint of the Event Notification.

When a 'normal' service wants to receive Event Notifications the above described scenario is not required. A 'normal' service runs within a server (such as Axis2 within Apache Tomcat) so all that is required of the service is to subscribe to the event with the delivery URI is its own endpoint.

The Savan module intercepts subscription messages for a service so it is not necessary to add anything to a service to handle subscriptions. When a service wishes to publish an event it makes a Savan API call with the event data as an argument. The Savan core code takes care of determining which subscribers are to be notified of the event and sends the event message to those subscribers.

2.2 Service Directory Service

The ACE Service Directory Service (ASD) stores location URL and interface information about currently running ACESOA services. Without the Service Directory, clients would have to 'know' the location (host, port and service name) and the interface (procedures and their arguments) of its desired services. This information would have to be hard coded or stored in configuration files for the client. If the interface or location of a service were to change the client code would have to be changed or the configuration file of the client would have to be changed. Changing client code would be impractical. Changing a configuration file would be cumbersome. With many different clients using the same service any service change would prohibitively difficult to manage.

ACESOA uses the [Web Service specification UDDI](#) (Universal Description Discovery and Integration) version 3 for a standard interface to service discovery. The implementation used is the [OpenUDDI Server](#) which is based on the Novell UDDI

Server. With UDDI v3, UDDI servers may be replicated and UDDI has the ability to register subscriptions for events.

The UDDI differs from the WFS (Web Features Service). The UDDI is intended to only store programmatic interface information about a service. The WFS is intended to store geographic features. The UDDI and WFS can be used together to locate and interface to a service in a specified geographic area. The UDDI can also be used to find the URL to the WFS.

Although there may be more than one Service Directory (UDDI) in an enterprise, generally a client or service will only use one Service Directory. There is no mechanism to 'search' for a Service Directory so the network location of an ASD for a client or service must be stored in a configuration file.

2.2.1 Service Publication

When an ACE service is starting up it must register its location and interface with the ASD. The Web Service specification for describing a service location URL and interface is WSDL ([Web Services Description Language](#)). In ACESOA WSDL v2.0 is used. Information from the WSDL for the service is stored in the UDDI server (ASD) as described in the OASIS Technical Note "[Using WSDL in a UDDI Registry](#)".

In addition to the 'standard' WSDL information for a service the ACESOA infrastructure uses Java's introspection to travel the service's Java class hierarchy to find each inherited class and each implemented interface. The name of each inherited class, interface and the class's name are stored in UDDI as added category values for the service's interface under the category key named 'acesoa:service-intf'. These categories are used by clients to search for services with a desired interface implementation. Example: All services that provide a temperature measurement implement the "org.tssn.service.sensor.temperature" interface. This interface name is stored in the ASD as a 'acesoa:service-intf' category for each service that implements it when the service is published by the ACE infrastructure as the service starts up. A client can ask the ASD for all services that have the 'acesoa:service-intf' category value of 'org.tssn.service.sensor.temperature'.

The mechanism for a service to register with the ASD is automatic and does not have to be considered by a service developer. ACESOA utilizes the Apache Axis2 implementation of SOAP with its AxisObserver interface. When Axis2 deploys a service the ACESOA implementation of an AxisObserver, named AceServiceEventListener, is invoked. The 'serviceUpdate' method of the AceServiceEventListener extracts the WSDL from the service and information from the service's configuration file and publishes the service's location URL and interface to the ASD(UDDI).

All of the information needed by the service to locate and communicate with the UDDI (username, password) is stored in the service's Axis2 service configuration 'service.xml' file. It is up to the administrator of the host running the service to ensure that the 'service.xml' file can not be read by unauthorized users of the host.

Since OpenUDDI (the implementation used for the ASD) utilizes Axis2 for its SOAP implementation, the SOAP message authentication and authorization mechanism used by ACE clients and services would be used with the UDDI. To avoid a chicken and egg problem the certificates for authentication signing of headers and messages are stored in a configuration file for the UDDI¹.

2.2.1.1 ASD Service Alive Status²

Previous version of ACE had a mechanism in the ASD and in the service infrastructure that provided a means of determining if a service that was registered with the ASD was still alive. The service would send a keep-alive message to the ASD periodically to tell the ASD that it was alive. The ASD had a timeout for each service. If the service did not send its keep-alive message to the ASD within the timeout period the ASD would remove the service information.

The UDDI has no ‘active service’ mechanism similar to past ACE ASDs. To have the UDDI only have published information of active services a ‘Active Status’ service is required. An Active Status service ‘pings’ each service listed in the ASD periodically to determine if it is alive. If the service fails to respond to N successive pings the Active Status service would un-publish the no longer responding service.

2.2.2 Service Discovery by Client

An ACESOA client uses the Service Directory service to discover the location URL and interface of one or more desired services. The ACESOA infrastructure provides an API for the client to call to perform the search. The method ‘ACEClient.findServices’ takes as an argument a string that is the name of a service interface that a service must implement in the UDDI search. The client must be coded to “know” how to interact with any service that implements the specified service interface. The ‘findServices’ method queries the UDDI (ASD) to search for all services that have published interfaces with categories that include the key name ‘acesao:service-intf’ and the category value as the specified service interface name. The UDDI returns the information it has on each service that matches the search.

The information returned by the ‘findServices’ method for each service found that matches the specified service interface includes:

- URL of the service
- Name of the service
- Namespace of the service (needed to create SOAP messages to the service).

¹ The mechanism for Authorization is has yet to be determined

² This service has not yet been implemented (November 7, 2007).

In a near future version of ACESOA, extended data stored in the UDDI for the service shall also be returned.

2.3 Authentication Service

NOTE: This infrastructure item is currently being integrated (November 7, 2007).

- Authentication service exists in each enterprise to provide certificates to clients to prove who the client user is.
- The authentication service verifies for a service the authentication certificates received by a service from a client.
- A trust relationship has to be established between enterprises so that Enterprise A will accept certificates issued by the authentication service of Enterprise B and visa-versa.

2.4 Authorization Service

NOTE: This infrastructure item has not yet been integrated (November 7, 2007).

- Services use authorization service to verify that the user specified in the client's message authentication certificate has the authority to invoke the procedure specified in the message
- The WS-Authorization specification has not been written yet.
- Items used to determine authorization include:
 - o Time of day
 - o Service name
 - o Host name
 - o GEO Location of service
 - o Name of user
 - o Procedure Argument values

2.5 Remote Services

NOTE: This infrastructure item has not yet been integrated (November 7, 2007).

- Remote services are not directly connected to the internet.
- Frequently have limited bandwidth and limited duration connections to the home office. (examples: GSM, Satellite Phone).

- Because of limited connections it is not practical to have remote services use the ASD, Authorization and Authentication services (too much communication overhead and out of date information).
- Use a service proxy that is connected to the internet and the wireless comm Link to the remote service.
 - o Proxy is responsible for handling ASD publishing, authentication and authorization of clients.
 - o Proxy communicates with remote service using single/fixed authentication and authorization. Remote service compares authentication and authorization with local files instead of using 'normal' authentication and authorization services.
 - o Comm link between proxy and remote service is only 'up' during message exchange.

3 Client and Service Development³

- The Apache Axis2 package is written in Java and thus clients and services developed for ACESOA are written in Java.
- Embedded services can use gSOAP and be written in C++.
- ACESOA has client and service Java code templates to ease development of new clients and services.

4 Scenario Examples

This section contains examples of how the ACESOA infrastructure, clients and services fit together.

4.1 Client/Service Messaging Example plus Remote Service Example

In this example an ACESOA client discovers a desired service that has published itself with the ASD. The service happens to be a Service Proxy for a remote service so the mechanism for communicating with a remote service is also shown in the example. The following interactions are described:

- Service publication

³ The content of this section will be written once the client/service authentication and authorization mechanisms have been implemented.

- Client search for a service via Service Directory (UDDI).
- Client user authentication.
- Client user authorization
- Client/Server communication
- Client/Remote Service communication
- Remote Service authentication
- Remote Service authorization

In this example the communication between a remote service proxy and a remote service is shown. This is different than 'normal' ACESOA communication since the assumption is that the communication link with the remote service is over a slow and time limited wireless link (example: GSM mobile phone).

Because of the limitations of the communication link the remote service keeps a local authentication file of trusted Certificate Authorities (CAs) and a local authorization file of credentials for a single user or only a very few users.

All communication with the remote service by clients must pass through the proxy service. The proxy service is the only entity that knows how to connect to the remote service and is the only user authorized to use the remote service (via the remote services authorization file).

In Figure 3 the initial state of the system is with the 3 ACESOA infrastructure services "Service Directory", "Authenticate" and "Authorize" and the remote service "Sensor" running.

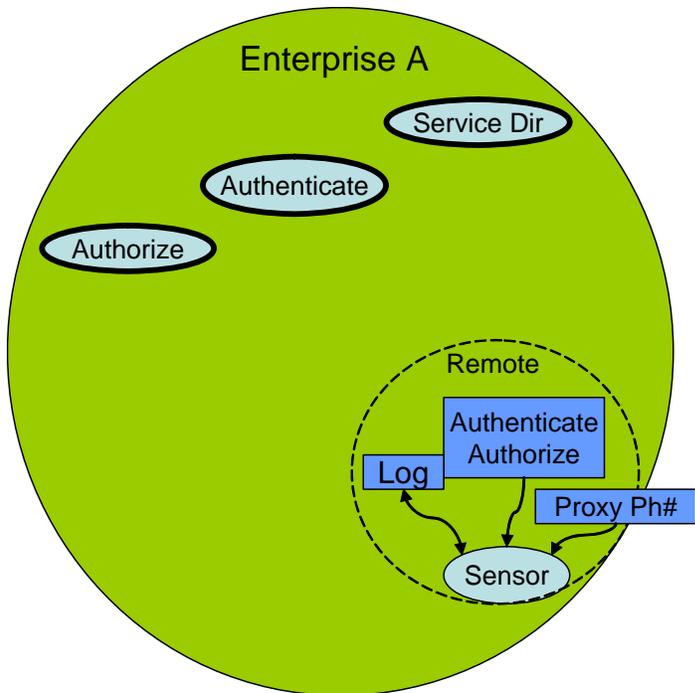


Figure 3 Client/Service Messaging Initial State

In Figure 4 the Sensor Proxy service starts up and is automatically published to the ASD by the ACESOA infrastructure. The service's location URL and interface information (WSDL information) is stored in the ASD UDDI database for later query by a client looking for the service.

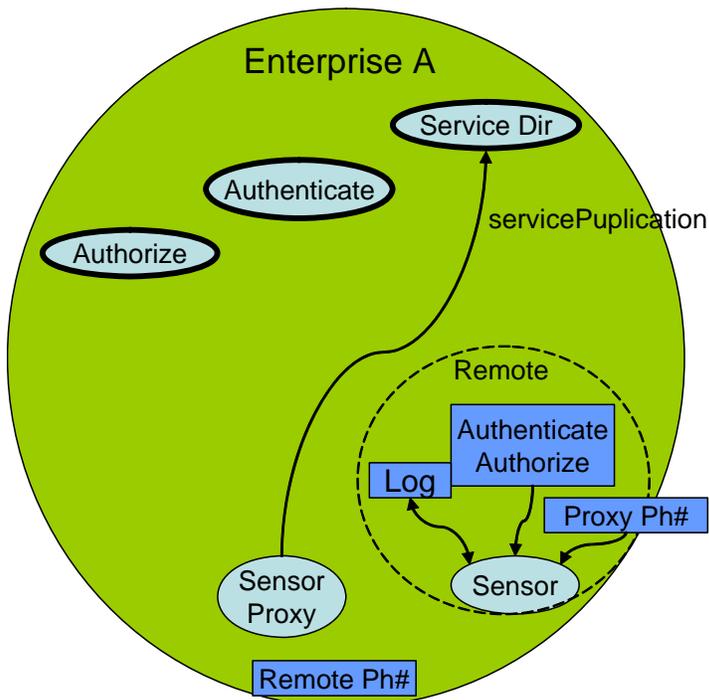


Figure 4 Service Publication

When starting up the Sensor Proxy Service obtains the information it needs about its remote service from its configuration file as show Figure 5. Included in the information is the remote service GSM phone number.

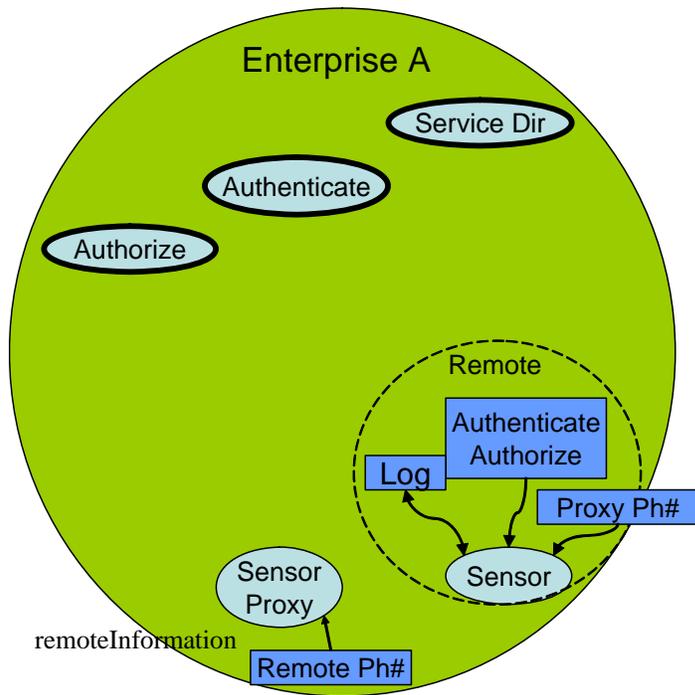


Figure 5 Proxy Service Configuration Info

In order to communicate with the remote service the Sensor Proxy service must act as a client and obtain the remote service client authentication token/certificate as shown in Figure 6. To simply the remote service, the token/certification for only Sensor Proxy user is stored in the Authentication file.

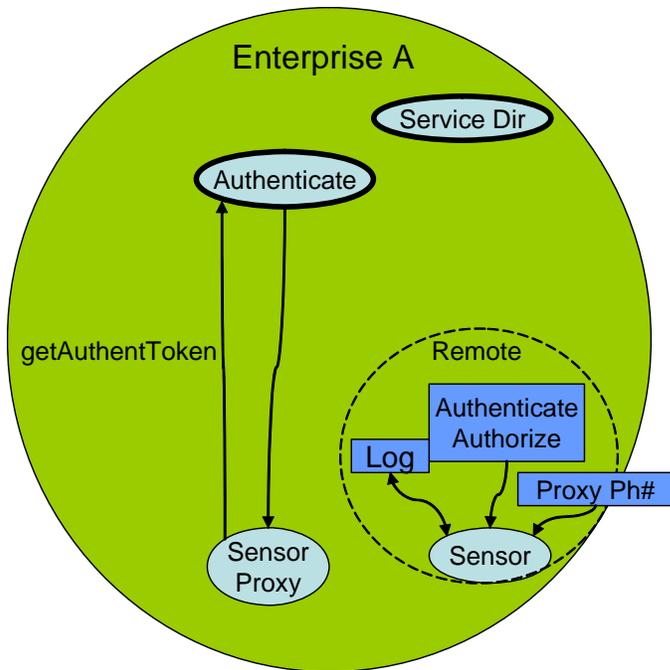


Figure 6 Remote Service Proxy Authentication Token

A client that intends to use the “Sensor Proxy” service starts up as shown in Figure 7. The name of the client is “Sensor Filter”.

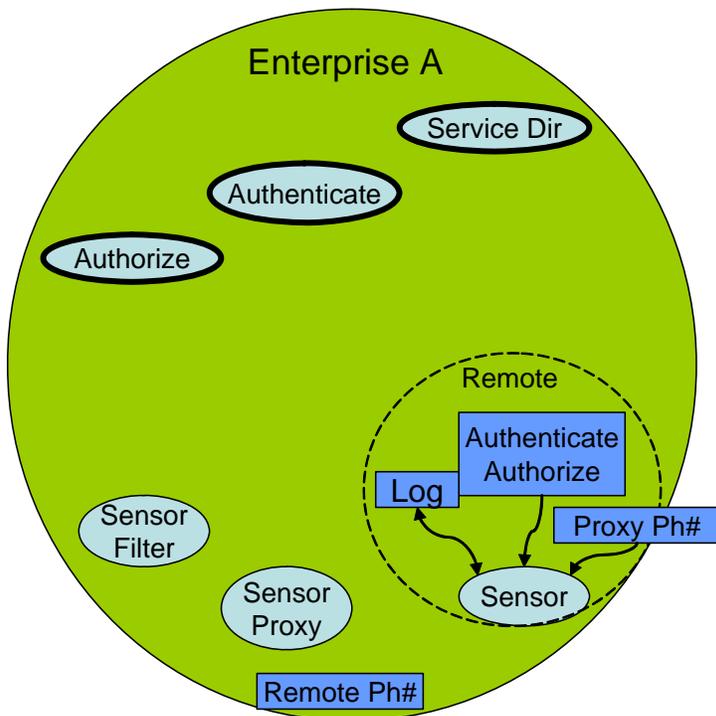


Figure 7 Client Startup

The client “Sensor Filter” wants to find the “Sensor Proxy” service as shown in Figure 8. The client knows the location of the ASD from an entry in its configuration file. An ACE API call is used by the client to perform the search query. The client specifies that it wants a service that implements the interface named “Sensor Proxy”. There could also be more query information such as the sensor type desired⁴.

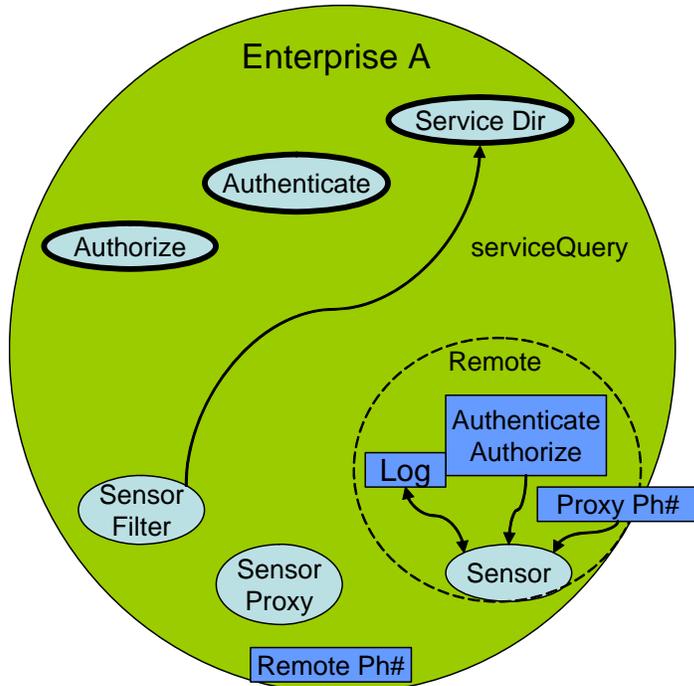


Figure 8 Client Query for Service

The ASD returns to the client those services that match the service query. In this example, shown in Figure 9, the service information returned to the client contains the WSDL information such as location (URL) and interface for the “Sensor Proxy” service.

⁴ The additional query information needed shall be reevaluated as more experience is gained with the sensor network.

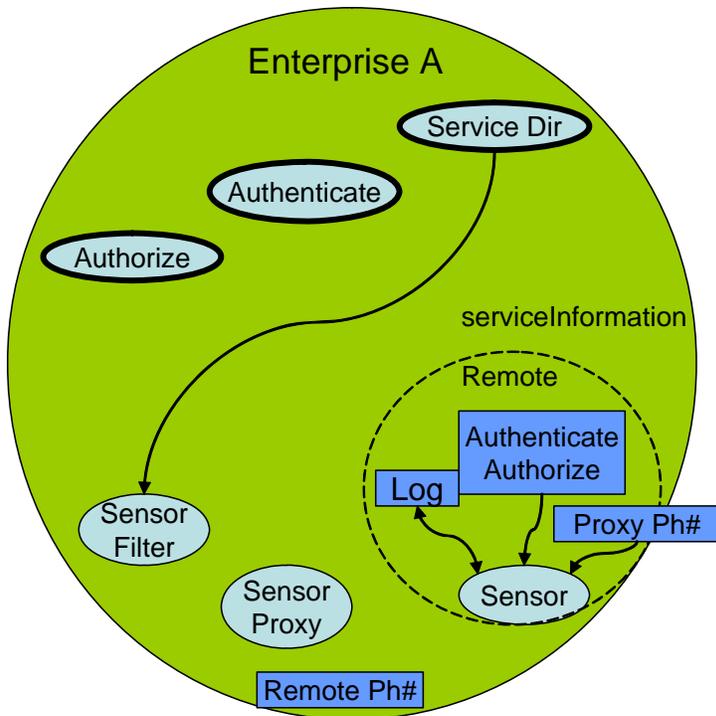


Figure 9 Client Query Return

In order for the service to recognize the client user the client must obtain an authentication token/certificate from a Certification Authority that the service trusts. In this case, as shown in Figure 10, the client queries the Enterprise A Authenticate service with a username and password for the client user. The message to the Authenticate service is encrypted by the ACESOA/Axis2 Rampart infrastructure so that the username and password are not easily seen on the network.

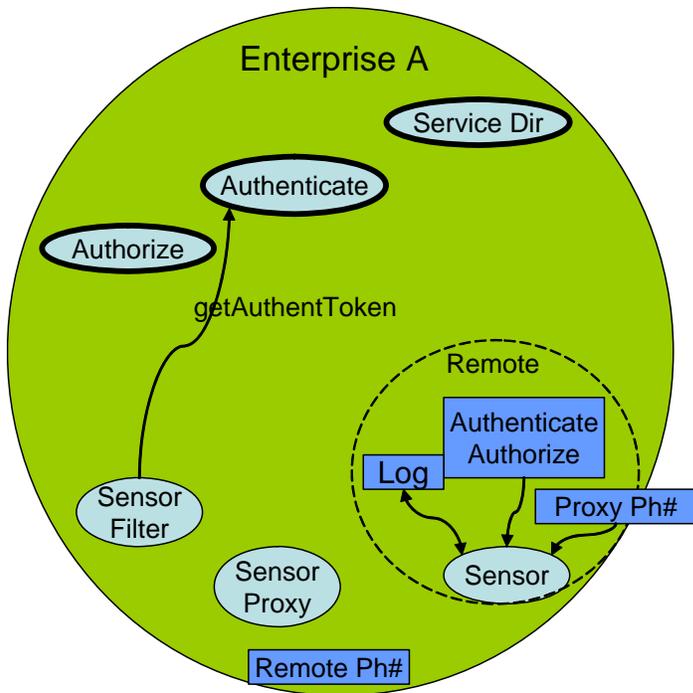


Figure 10 Client gets Authentication Token

Figure 11 shows the Authenticate service returning an authentication token/certificate after it has verified that the username and password provided by the client is valid.

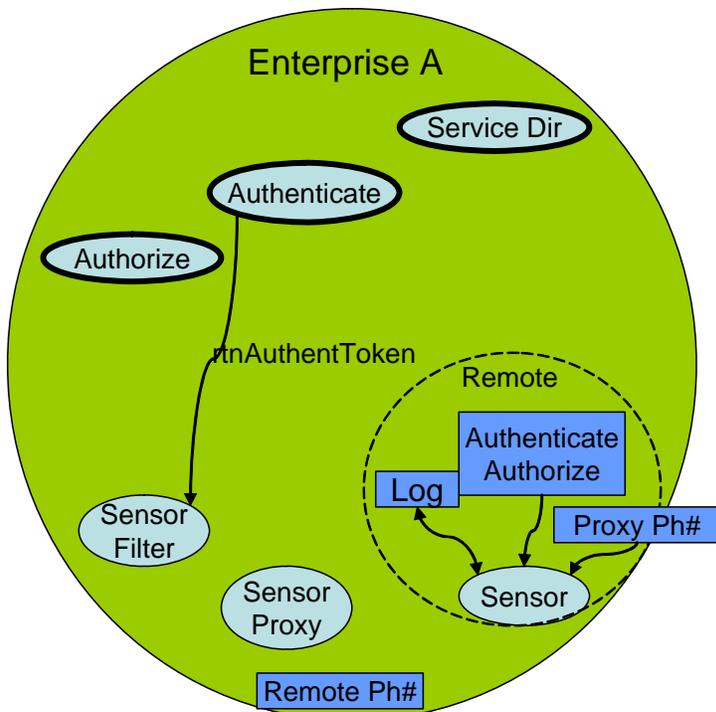


Figure 11 Client Authenticate Token Return

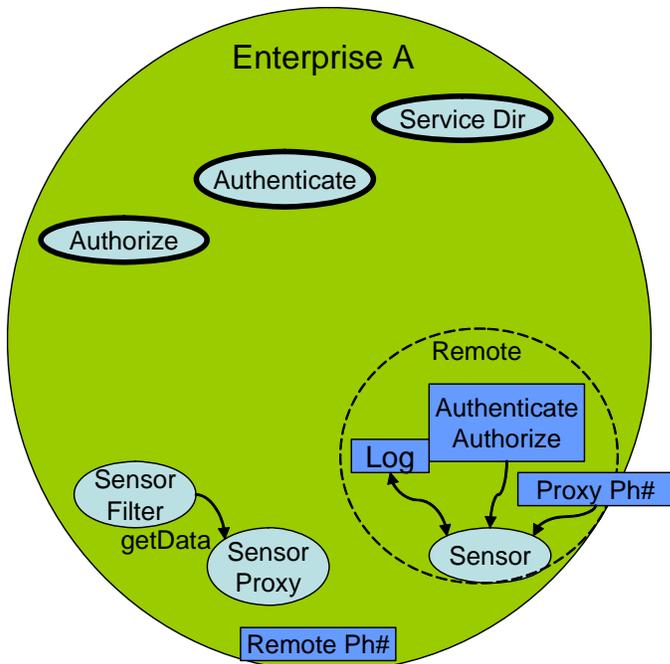


Figure 12 Client Request of Data from Service

The client, using the WSDL information obtained about the service from the ASD, composes and sends a data request to the service. The ACESOA/Axis2 infrastructure handles signing the message (authentication info) and encrypting the message before it is sent to the service. See Figure 12.

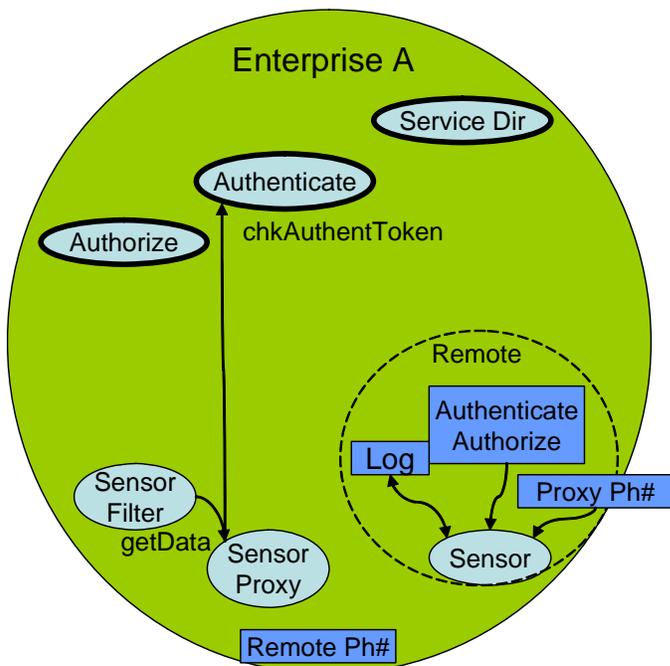


Figure 13 Service Authentication Check

The ACESOA/Axis2 infrastructure checks the authentication token/certificate (signature) of the message to determine if it is authentic and from a trusted Certificate Authority. First the service infrastructure looks to see if the information for this user is already in the service authentication cache. If it is not then it sends a request to the ACESOA Authenticate service to verify the users certificate/token. See Figure 13.

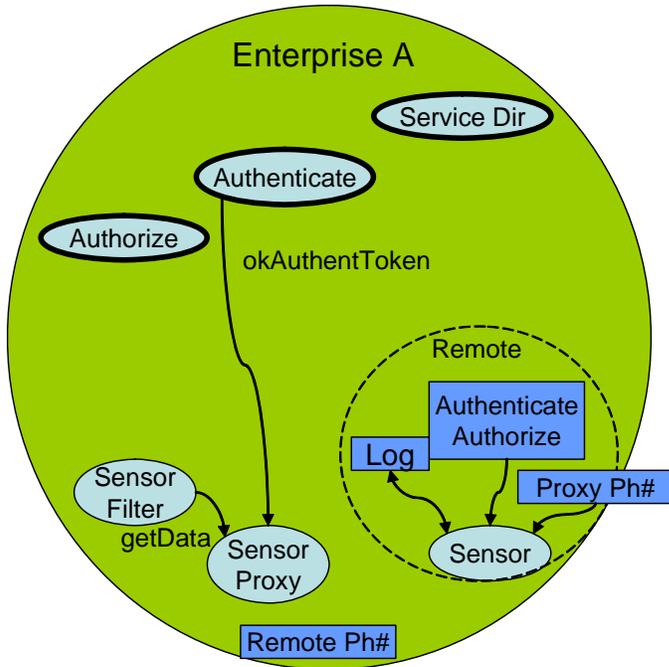


Figure 14 Verified Authentication

If the authentication token/certificate that the “Sensor Proxy” sent to the Authenticate service was valid then the Authenticate service returns an “OK” to the “Sensor Proxy” service. To save time in the future the ACESOA infrastructure saves the client user’s authentication token/certificate in a local authentication cache so that the Authenticate service will not have to be queried for the next message from the same user. The authentication entry in the cache is given a timeout period. The entry in the cache is removed if the timeout occurs.

Note shown in Figure 14 is that the ACESOA infrastructure for the “Sensor Proxy” service subscribes to the Authenticate service so that the “Sensor Proxy” service is notified if the trustworthiness of a Certificate Authority changes.

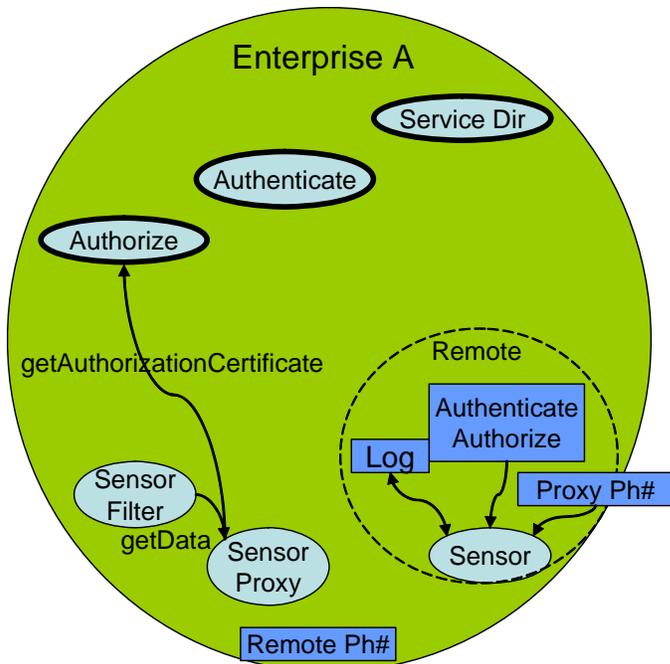


Figure 15 Service Authorization of User

In Figure 15 the ACESOA is querying the Authorize service to determine if the user that has just been authenticated is authorized to invoke the procedure specified in the message. To save time the infrastructure provides an authorization cache. The authorization cache is first examined for a certificate/credential for the client user before trying the Authorize service.

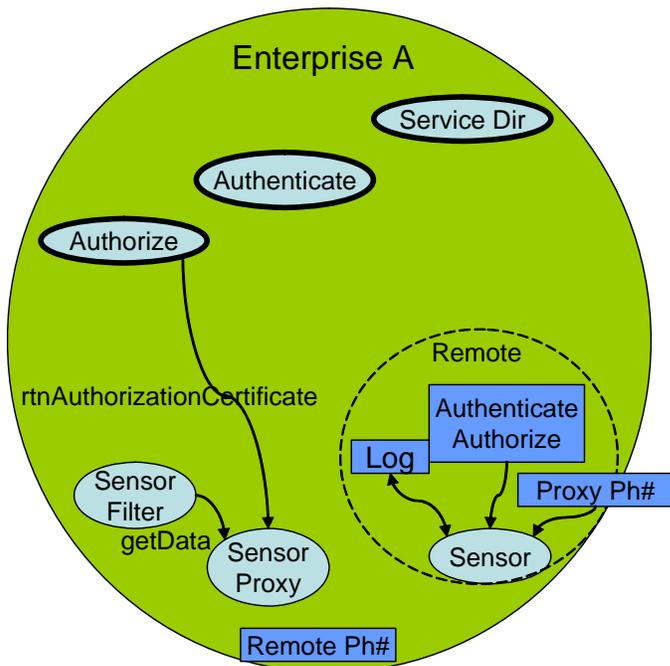


Figure 16 Authorization Certificate/Credential

An authorization certificate/credential is returned by the Authorize service to the requesting service as shown in Figure 16. This certificate/credential is stored in a local cache by the ACESOA infrastructure so that the Authorize service will not have to be queried for the next message from the same client user. A timeout period is set for the cache entry. At the end of the timeout period the entry is removed from the cache.

Not shown in the figure is the subscription of the “Sensor Proxy” service to the Authorize service by the ACESOA infrastructure so that the service can be notified of any change in the authorization certificate/credential of the client user.

In Figure 17 the remote service Sensor is the service that does the actual measurements so the Sensor Proxy must connect to the remote service. The Sensor Proxy uses the phone number it obtained from its configuration to call the remote sensor using its attached GSM phone (not shown).

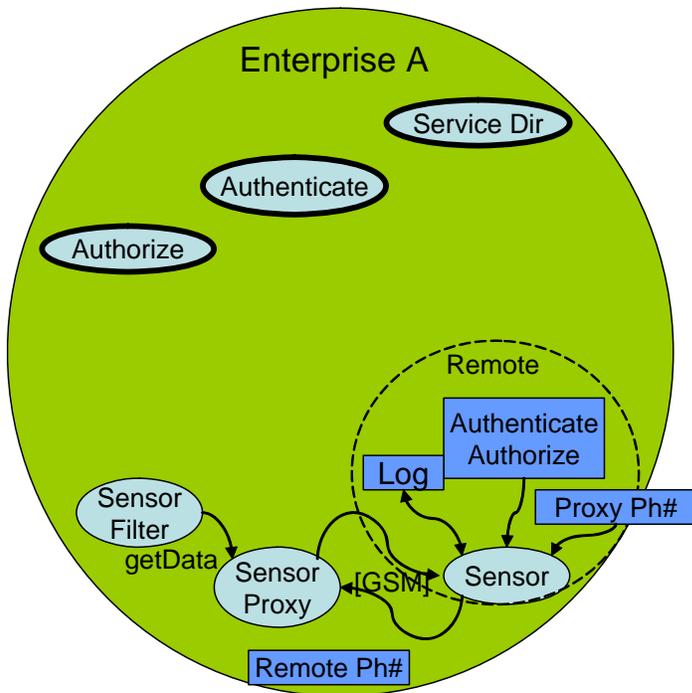


Figure 17 Remote Service Connection

The Sensor Proxy service sends the request message ‘getData’ to the remote Sensor service. The ACESOA/Axis2 infrastructure includes the Sensor Proxy user’s authentication token/certificate in the message and encrypts the message. This is shown in Figure 18.

The remote service compares the certificate authority in the message’s authentication certificate/token and compares it with the trusted CA in its local Authenticate file. If the CA is trusted then the remote service checks to see if the Sensor Proxy user is allowed to invoke the ‘getData’ procedure by evaluating the credentials in the local Authorize file. If the user is authorized then the remote Sensor service obtains the measurement data.

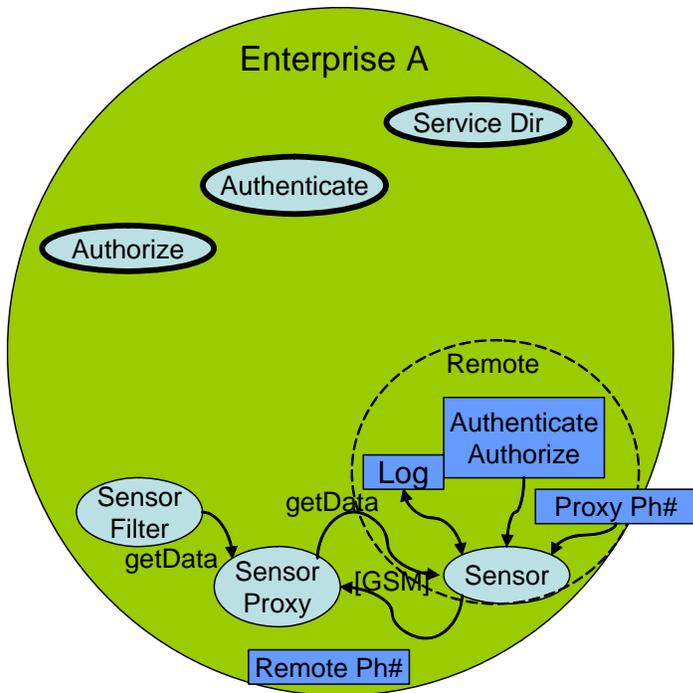


Figure 18 Remote Service Data Request

The remote service returns the data as shown in Figure 19. The ACESOA infrastructure encrypts the message. The Sensor Proxy service then passes the data on to the client that originally requested the data.

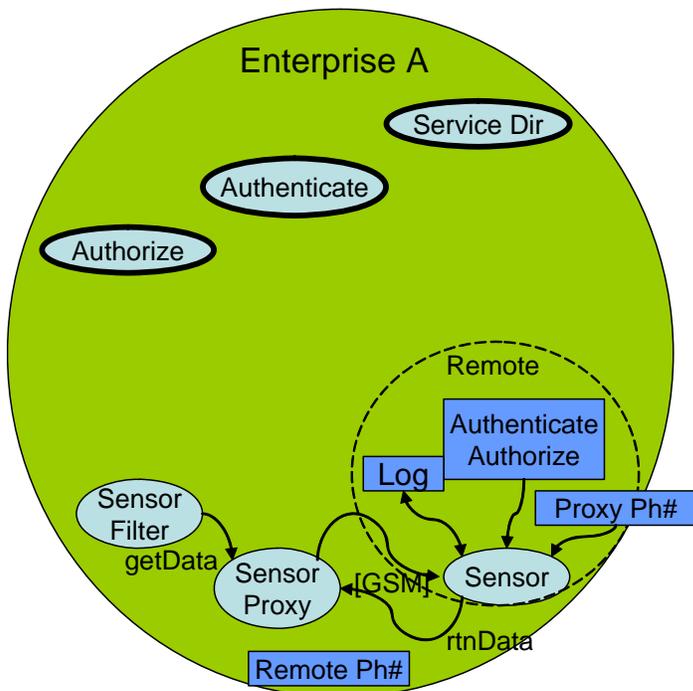


Figure 19 Remote Service Data Return

The Sensor Proxy tears down the GSM connection once the message transaction has completed as shown in Figure 20.

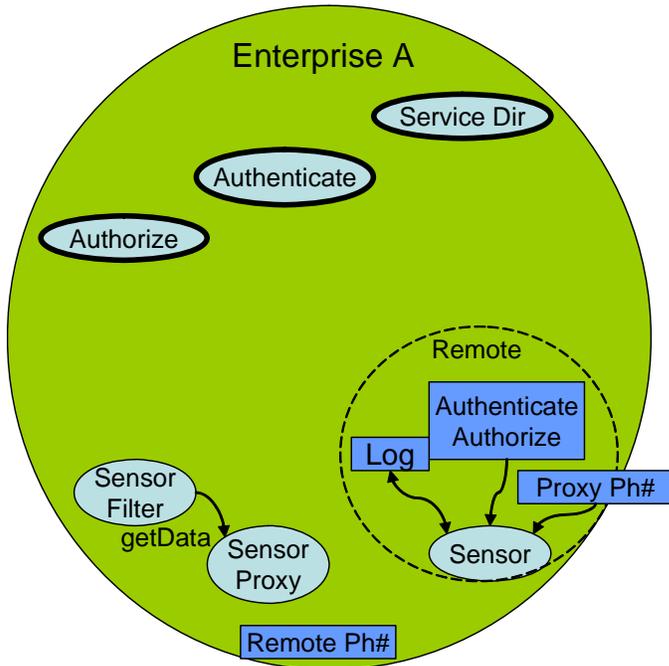


Figure 20 Remote Service Connection Termination

In Figure 21 the measurement data is finally returned to the client.

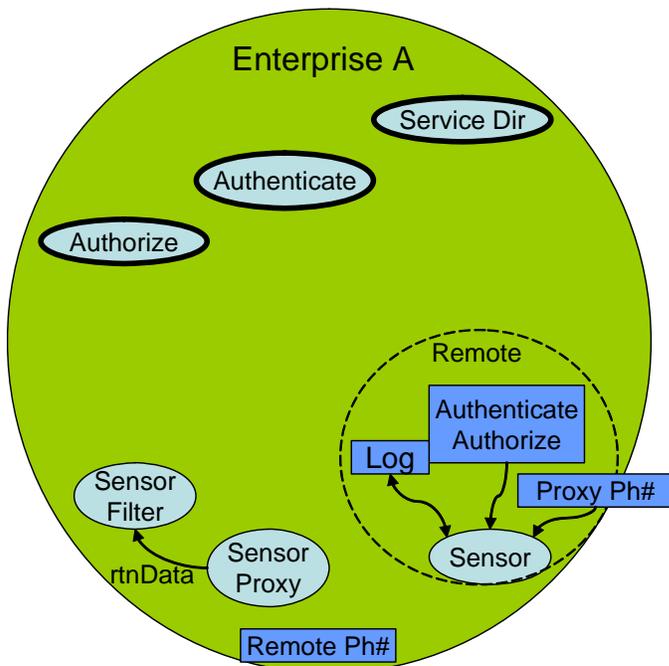


Figure 21 Client Data Return

4.2 ACESOA Federation Example

In this example a client in Enterprise B wants to query a Sensor DB service in Enterprise A. The majority of the interactions of clients and services are shown in this example. The following interactions are described:

- Inter-Enterprise authentication trust.
- Inter-Enterprise service publication subscription
- Service publication
- Client search for a service via Service Directory (UDDI).
- Client user authentication.
- Client user authorization
- Client/Server communication

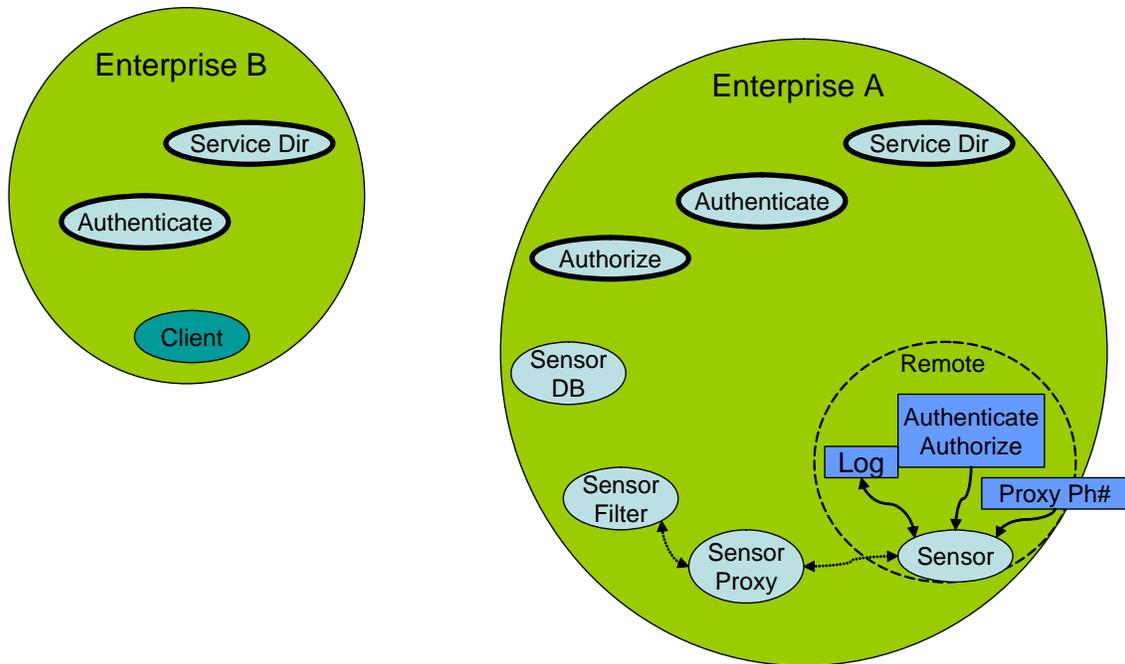


Figure 22 Federation Initial State

In Figure 22 Enterprise A has its ACE Service Directory, Authenticate and Authorize service running. It also has a remote sensor that only communicates with the Sensor Proxy Service. The sensor data from the Sensor Proxy service is manipulated by the Sensor Filter service to be more useable by sensor clients.

For this scenario the service Sensor DB start up is shown. The purpose of the service is to store sensor data in a database and provide the sensor database data to clients inside and

outside Enterprise A. It will obtain its sensor data from the Sensor Filter service although the “Sensor DB”-“Sensor Filter” interaction is not shown in the example.

At some point in the future the client in Enterprise B will want to obtain data from the database in the Sensor DB service of Enterprise A.

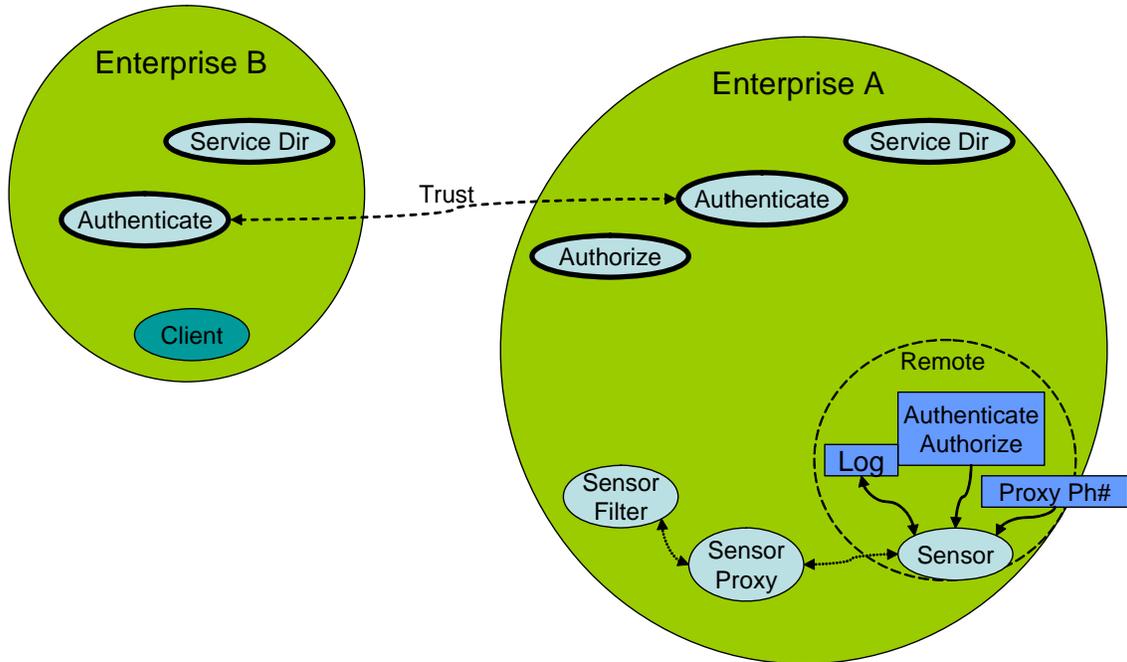


Figure 23 Federation Authentication Trust

In order for a client in Enterprise B to authenticate itself to Enterprise A there must be a trust relationship between the Authenticate services of enterprise A and B (see Figure 23). This must be done by humans in the two enterprises by exchanging Certificate Authority signatures over a secure communication mechanism (example: encrypted and digitally signed email). In this example an administrator in Enterprise A takes the Certificate Authority signature that it has received from an administrator in Enterprise B and stores it in a database of trusted Certificate Authorities for the Authenticate service of Enterprise A.

In Figure 24 the ASD of Enterprise B is sending a subscription request to the ASD of Enterprise A. This subscription request tells the ASD of Enterprise A that the ASD of Enterprise B should be notified of any services that publish with Enterprise A’s ASD. This reduces internet traffic since clients query their own ASD to find services in other enterprises.

The ASD of Enterprise A may also subscribe to the ASD of enterprise B but that is now shown in this example.

An enterprise will want to keep some services private (intra-enterprise). In this case there would be two ADSs in the enterprise. One ASD would be for services that allow inter-

enterprise usage. The other ASD would be for service with intra-enterprise usage only (not shown).

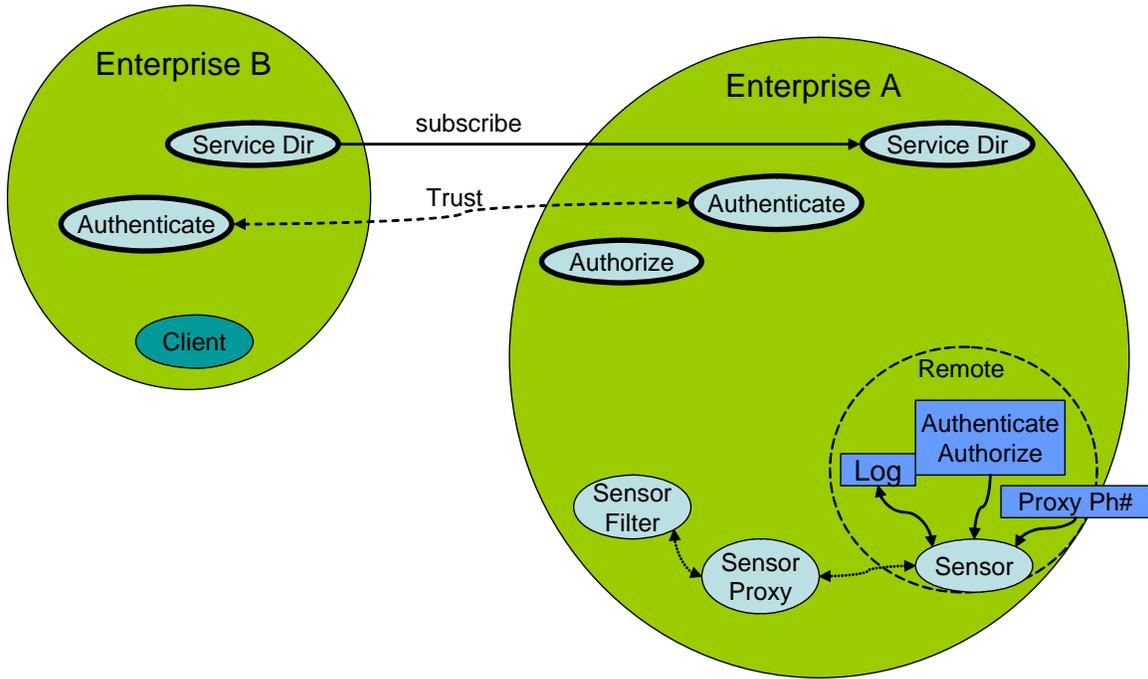


Figure 24 Federation ASD subscription

In Figure 25 the Federation A service ‘Sensor DB’ has been started. The startup could have been initiated by a host computer booting up or a human could have started it manually.

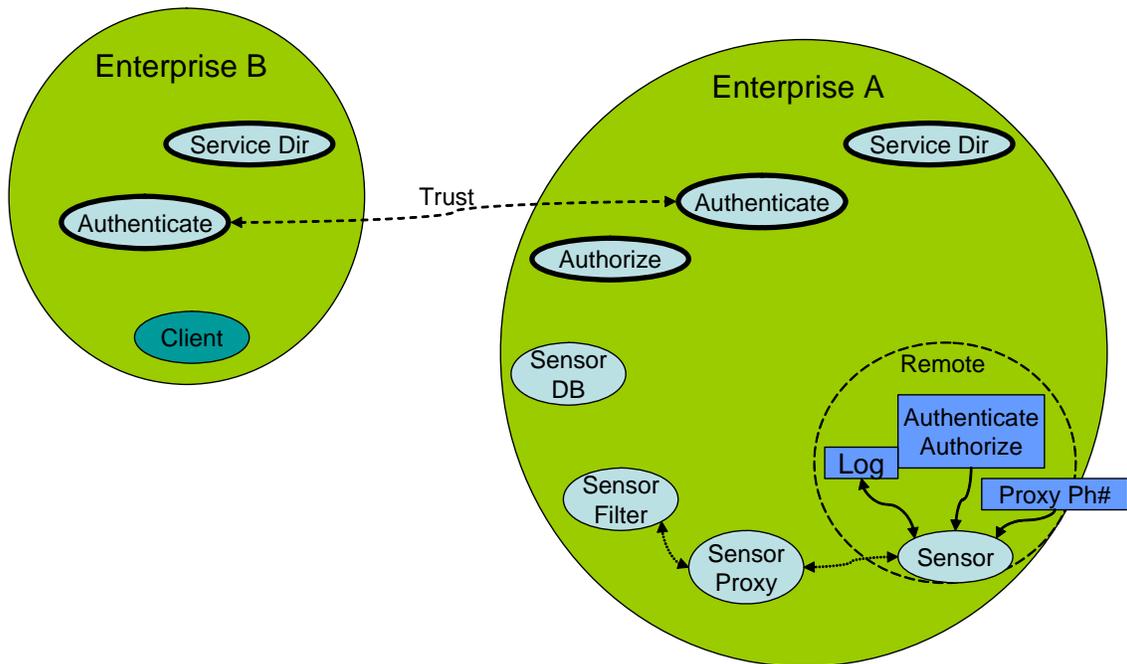


Figure 25 Federation Sensor DB Service Startup

In Figure 26 the service Sensor DB publishes its internet location information and interface to the public (inter-enterprise) ASD so that clients outside Enterprise A are able to find it.

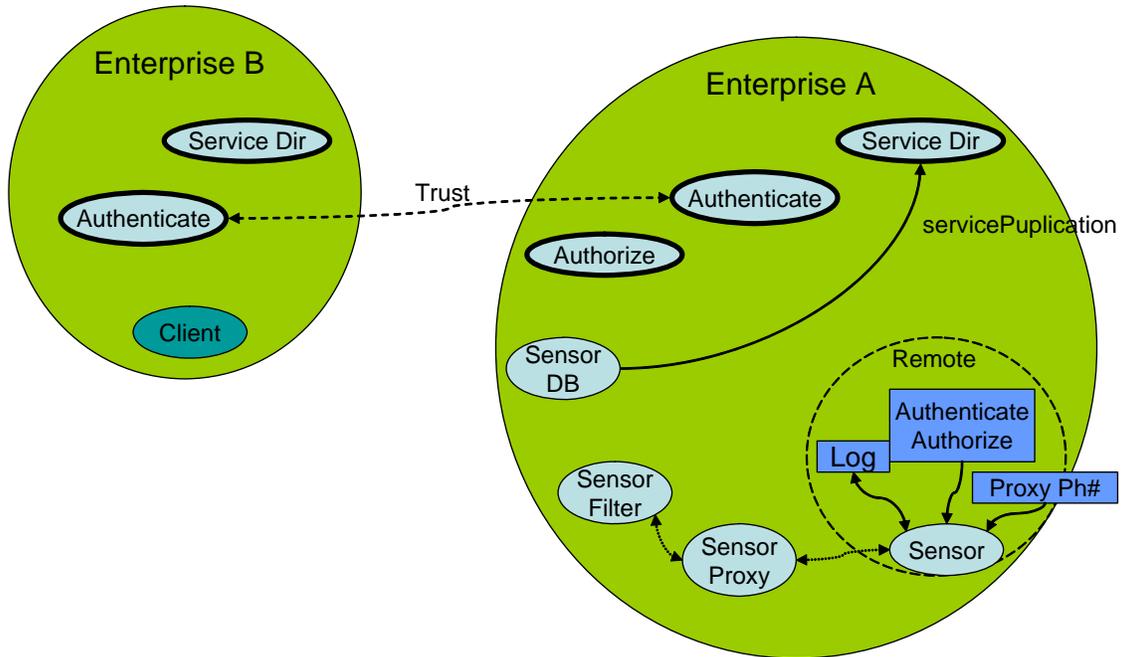


Figure 26 Federation Service Publication

In Figure 27 the ASD of Enterprise A has received the publication of the Sensor DB service. For each ASD that has subscribed to it, Enterprise A's ASD publishes the information for the Sensor DB service. In this example it publishes the service information to Enterprise B's ASD since it had previously subscribed to Enterprise A's ASD.

Although not shown in the figures, service Sensor DB queries the ASD of Enterprise A for the Sensor Filter service. The ASD returns the location and interface information for the Sensor Filter service and the Sensor DB service connects as a client to the Sensor Filter service to obtain sensor readings to store in the Sensor DB database.

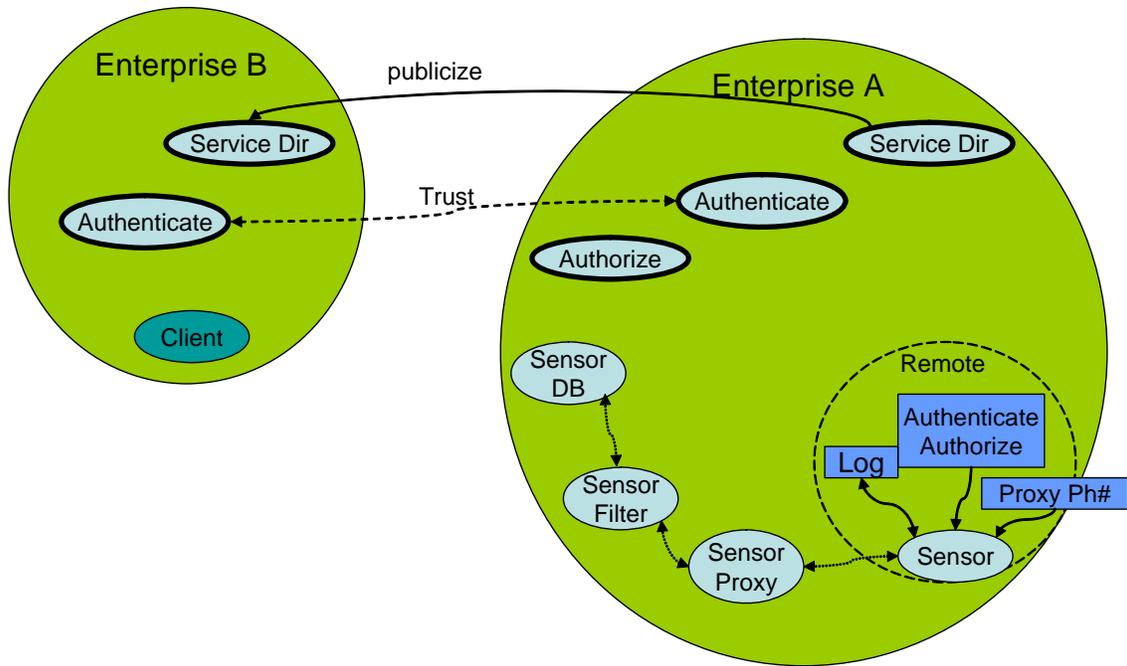


Figure 27 Federation Inter-Enterprise Publication

The client in Enterprise B wants to find and query the Sensor DB service in Enterprise A as shown in Figure 28. The client sends a service query request to its Enterprise B ASD looking for services that implement the Sensor DB interface.

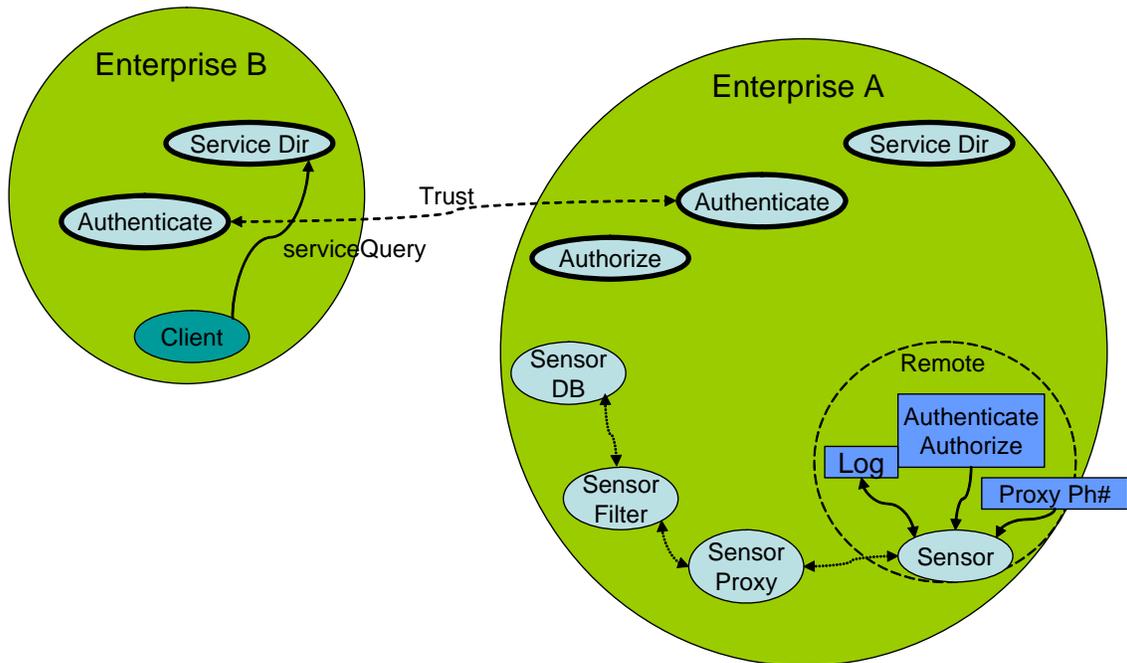


Figure 28 Federation Client Query for Service

In response to the clients query for service information, Enterprise B's ASD returns the location URL and interface information for the Sensor DB service that it knows about in Enterprise A. This is illustrated in Figure 29.

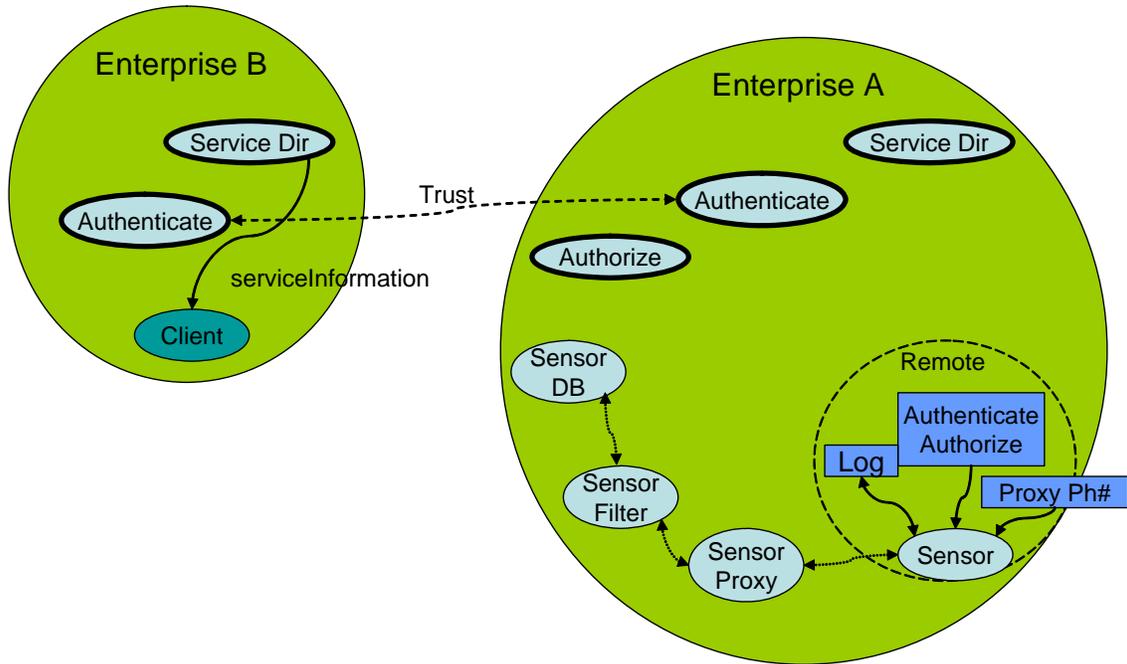


Figure 29 Federation Client Query Service Information

The client must authenticate itself with the Sensor DB service in Enterprise A. To do this it must have a certificate or token that will be accepted by Enterprise A. Since the trust relationship has been established between the Authenticate services of Enterprise A and Enterprise B, the client can request its certificate/token from its Authenticate service as shown in Figure 30. In the token request to the Authenticate service, the client provides a username and encrypted password for identification of the client user.

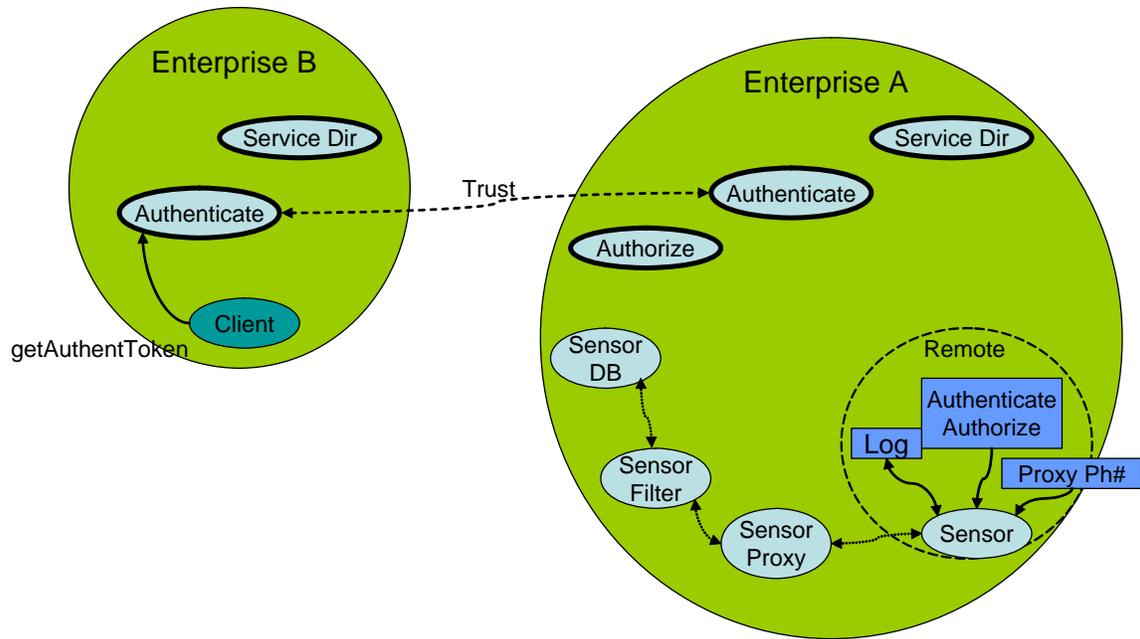


Figure 30 Federation Client Authenticate Request

The Authenticate service, once it verifies the validity of the username and password, returns a certificate/token to the client to use to authenticate itself with services. This is shown in Figure 31.

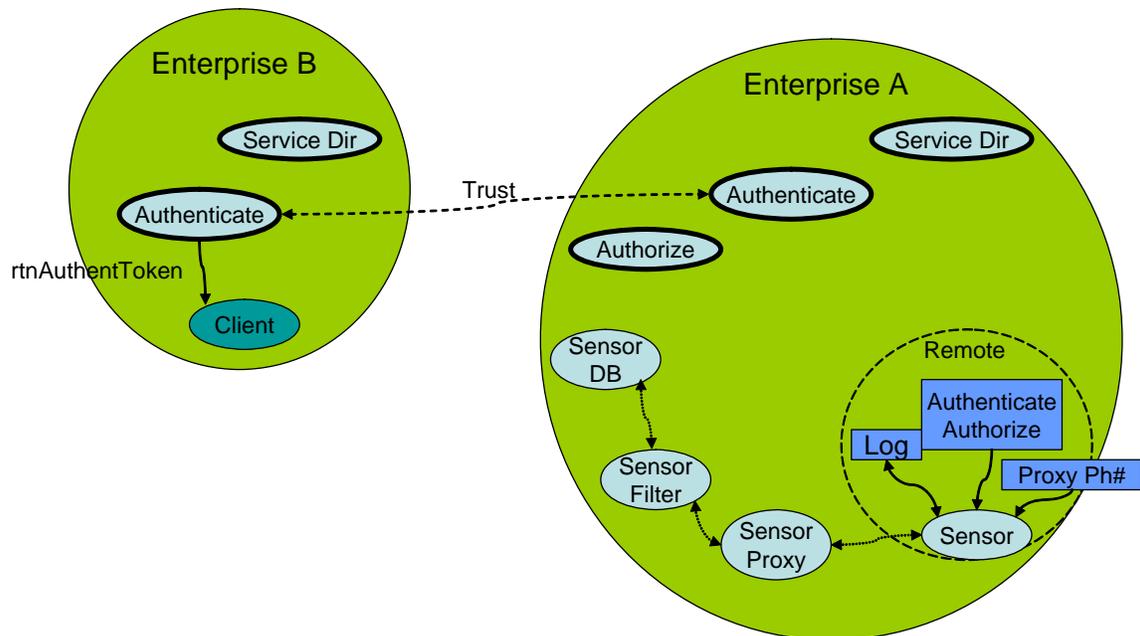


Figure 31 Federation Client Authenticate Return

As shown in Figure 32, using the network location information returned by the ASD, the client creates a message containing the location URL and name of the service and adds the procedure request 'getData'. The authenticate token is automatically added to the

message and the header and body of the message are encrypted by the ACESOA infrastructure and Apache Rampart Axis2 module.

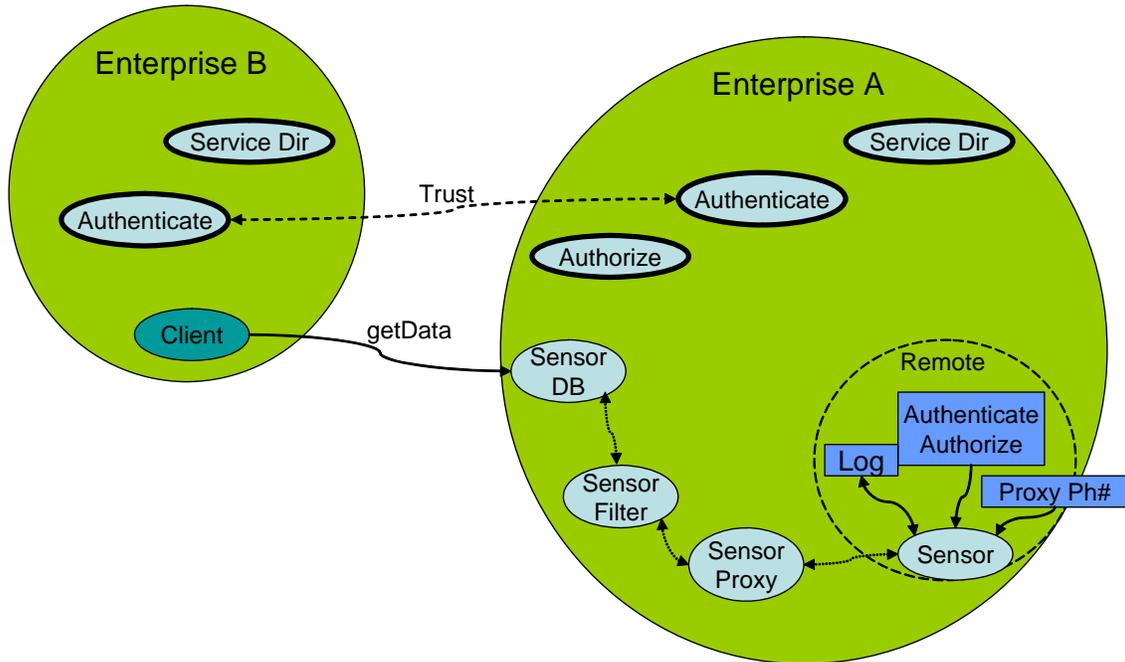


Figure 32 Federation Client Request of Service

The service Sensor DB has received the 'getData' request from the client as shown in Figure 33. The ACESOA infrastructure first checks the local authentication cache in the service to see if the client user is already known, if the user is not known then the Enterprise A Authenticate service is queried to authenticate the client user by checking the user's token/certificate to see if it was issued by a trusted Certificate Authority.

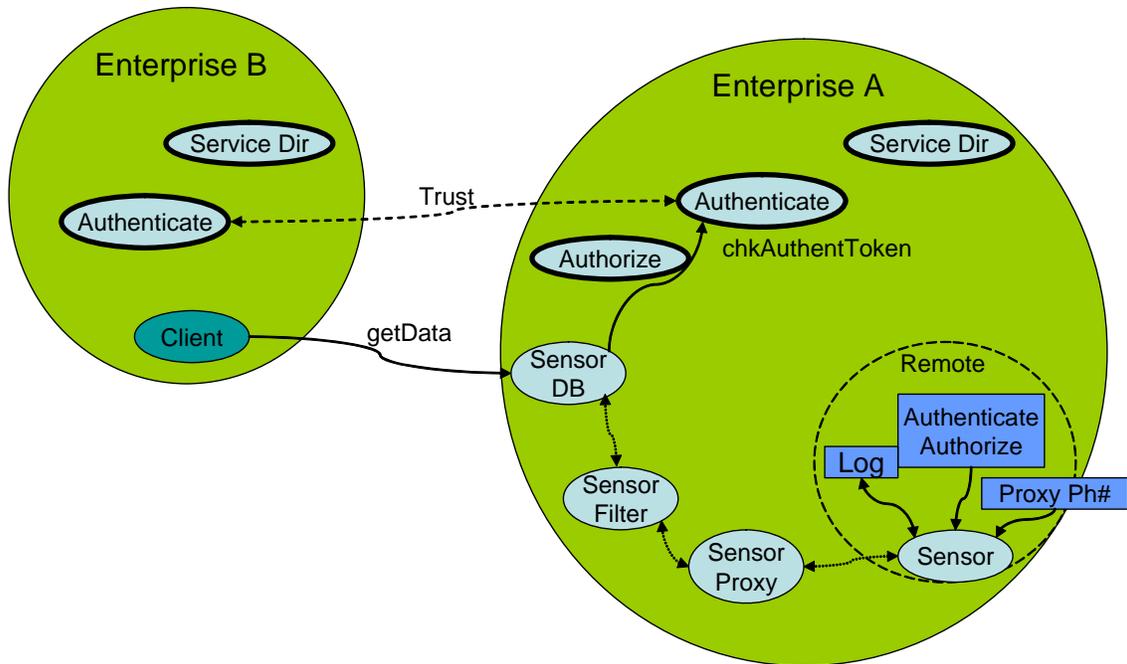


Figure 33 Federation Authenticate Client

For this scenario, as shown in Figure 34, the Authenticate service tells the Sensor DB service that the user is really who he/she claims to be. The Sensor DB stores the user's authentication information in a cache to be used for future queries by the same user to save time by skipping the authentication with the Authenticate service. The entry in the cache is given a limited lifetime and then removed from the cache at the end of the lifetime.

Not shown in the figure is that the Sensor DB has subscribed with the Authenticate service to be notified if a Certificate Authority is no longer trusted. When this notification happens all certificates/tokens cached by the service that were issued by the no longer trusted CA are removed from the cache.

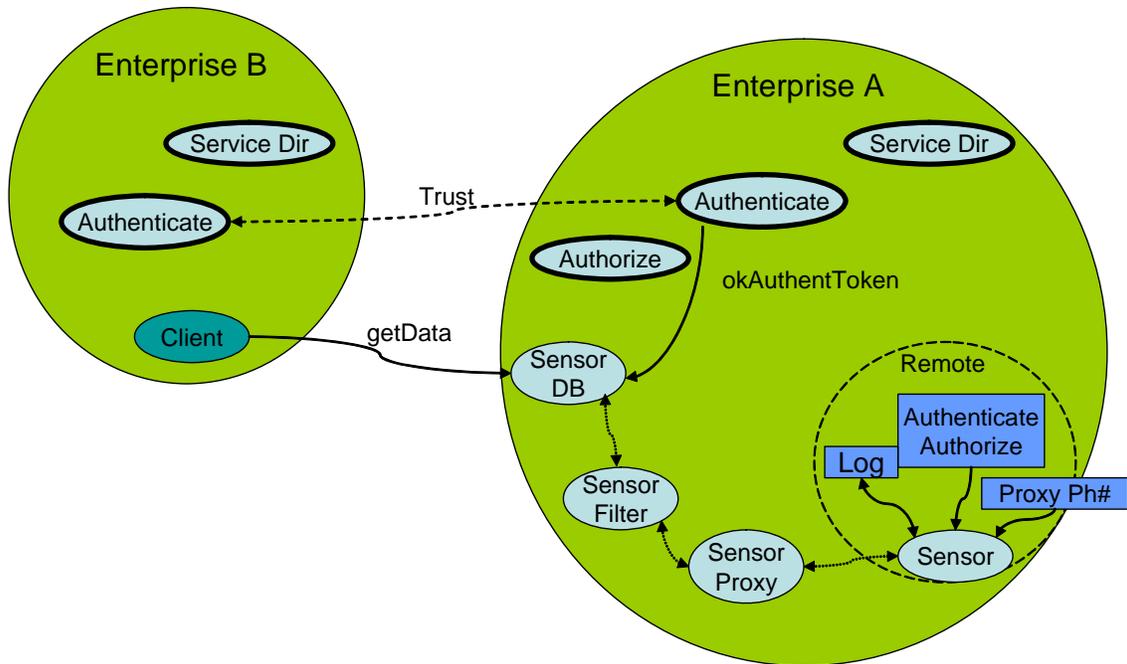


Figure 34 Federation Authenticate OK Result

Once the client user has been authenticated the service has to determine if the user is allowed to invoke the requested operation 'getData'. The Sensor DB sends a query to the Authorize service to get the authorization credentials/certificate of the user if the service does not already have the credential in a local credential cache. This request is shown in Figure 35.

Note shown in the figure is that the Sensor DB has subscribed to the Authorize service to be notified when credentials change.

The credentials in the Authorize service have to have been setup by an administrator some time in the past.

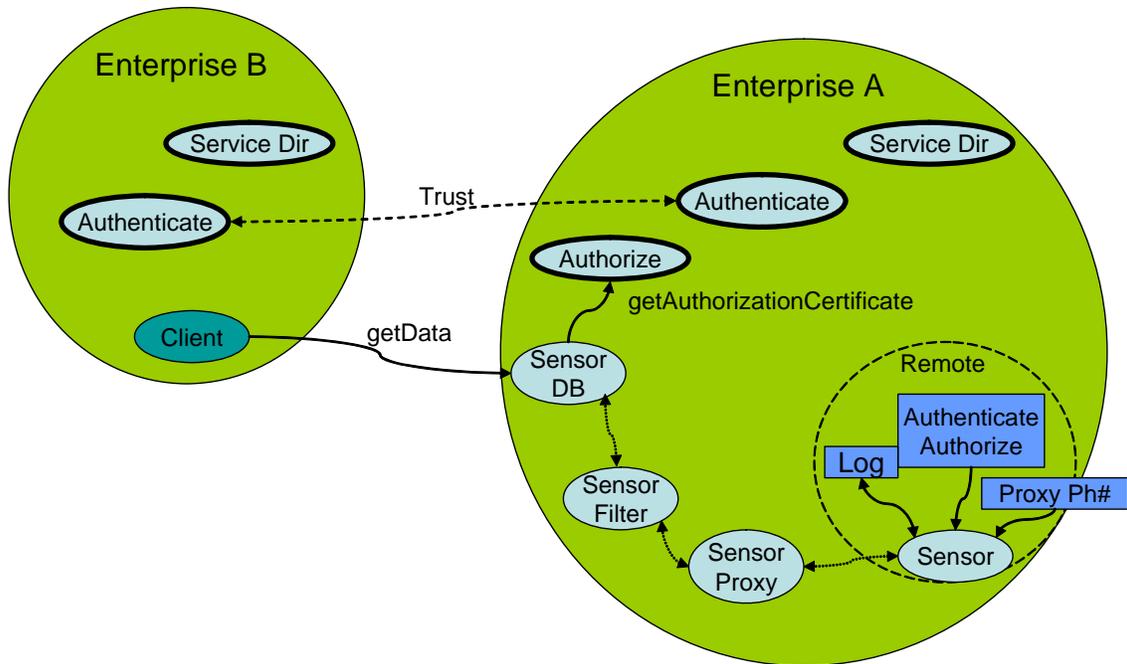


Figure 35 Federation Get Authorization

Figure 36 shows the returned of a user's authorization certificate/credential by the Authorize service to the Sensor DB service. The Sensor DB service caches the certificate/credential to use with future requests by the users until the cache entry times-out or the Authorize service notifies the service of a change in authorization certificate/credential.

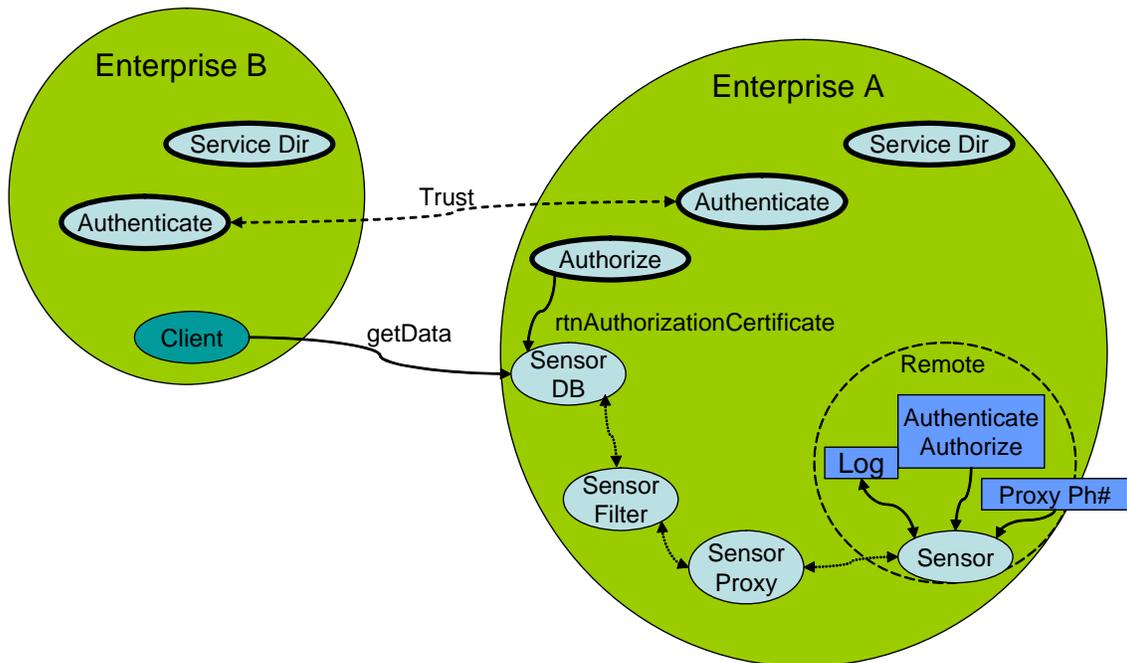


Figure 36 Federation Returned Authorization

The final act in a client from Enterprise B requesting data from a service in Enterprise A is for the service to return the requested data as shown in Figure 37. The ACESOA/Axis2 infrastructure encrypts the return message and sends it to the client then it is decrypted and given to the client code.

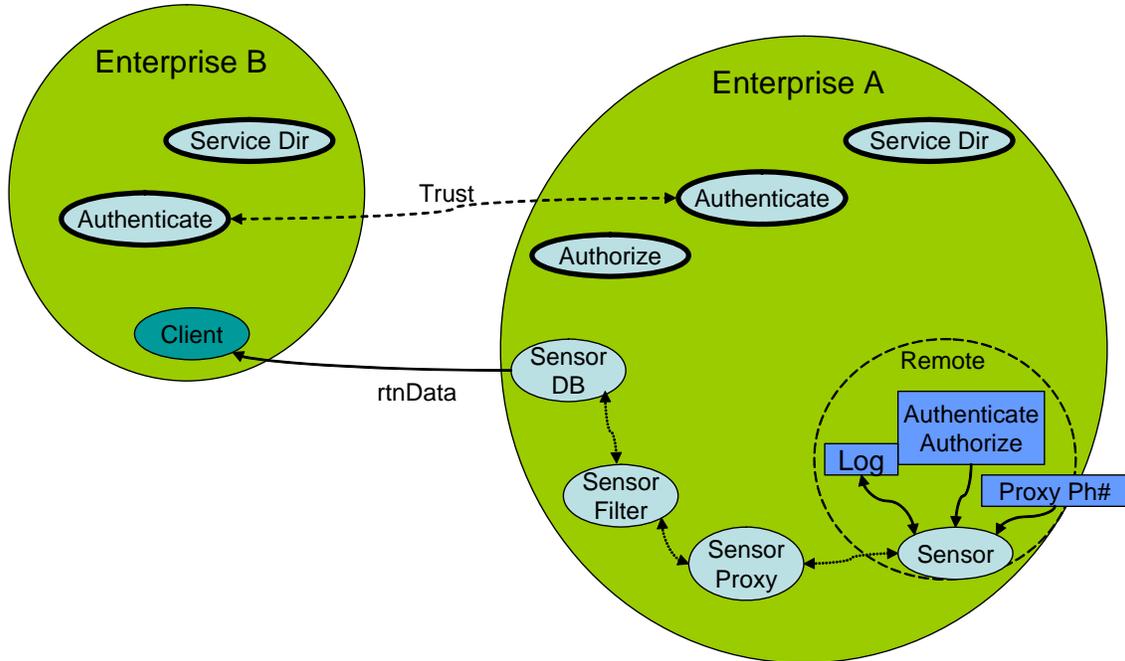


Figure 37 Federation Service Data Return to Client

4.3 Event Notification Examples

The following examples illustrate the messaging that occurs to subscribe to, publish and consume Event Notifications. Note that in these examples it is assumed that service discovery, authentication and authorization are occurring but is not shown.

The WS-Event specification implementation used in the ACE environment is Savan. Savan provides an API for clients to send subscription messages to an eventing service. Savan intercepts subscription messages coming into an eventing service so no external API is needed to receive subscriptions. Services use a Savan API to publish events. There is no API to receive published event notification messages.

4.3.1 Service – Service Notification

In Figure 38 the messages involved with having one service subscribe to and be the consumer of Event Notifications of another service are shown. This is the simplest Event Notification scenario.

Message 1 is the Sensor Filter Service using the client Savan API to send an Event Notification subscription to the service Sensor. Included in the subscription is a filter to determine which events the consumer is interested in. Also included is the ReplyTo: field

which is the URI of the endpoint to send Event Notifications to for this subscription. In this case the ReplyTo is the Sensor Filter service endpoint.

The Savan WS-Eventing Axis2 module intercepts the subscription in the Sensor service and stores the subscription information (hidden from the service Sensor business implementation).

When an event occurs (message 2 originating from some other source) the service Sensor business implementation places the event data into a data binding (class) and then calls the Savan PublishEvent API with the event data. The Savan implementation then handles checking each subscription filter to determine which consumers are to be sent Event Notifications. The subscriptions that pass the filter check then have the event data sent (published) to the ReplyTo: found in the subscriptions (message 3 in Figure 38).

Sensor Filter receives the event notification as a ‘normal’ operation message.

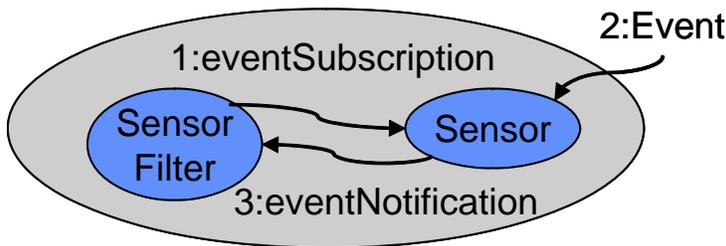


Figure 38 Service - Service Event Notification

4.3.2 Client - Service – Service Notification

This scenario (Figure 39) involves a client subscribing to events in a service on behalf of another service. This is a common scenario where a user is using a client application to tell a service where to send Event Notifications.

In this case the eventSubscription message ReplyTo field from the client contains the URI of the Sensor Filter service. See section 4.3.1 for more details of the messages and the implementation tasks of the remainder of the messaging.

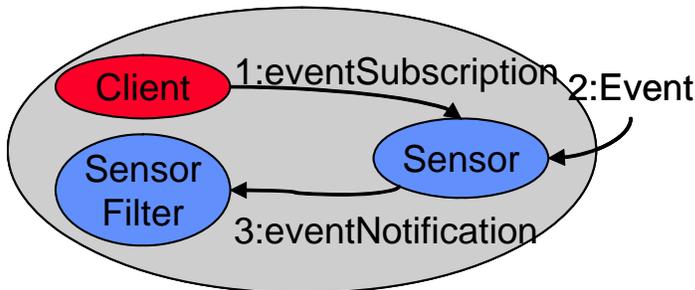


Figure 39 Client - Service - Client Notification

4.3.3 Client – Service Notification

In the scenario shown in Figure 40 the desire is to have Event Notifications delivered to a client. Clients can not receive notifications directly (as discussed in section 2.1.3). This restriction requires a service to be embedded in the client to receive the notifications. The embedded service then passes the notification data to the client.

In order for a service to run it must be handled by a server. In this scenario the client starts an embedded server (1: in the figure), specifying the name of a desired service to start. The server then starts the specified embedded service.

The client registers a callback class (usually itself) with the embedded service to be used when a notification is received by the embedded service (2: in the figure).

Messages 3:, 4: and 5: in the figure are the same as described in section 4.3.2 except that the ReplyTo: field of the subscription message is the URI endpoint of the embedded service. The client obtains the embedded service endpoint from from the server.

When the Embedded Service receives the notification it looks up the callback class (previously registered by the client) associated with the type of notification data received. It then calls the callback method of the callback class with the notification data as the argument (6: in the figure). If the callback class was the client class then the client directly receives the notification data.

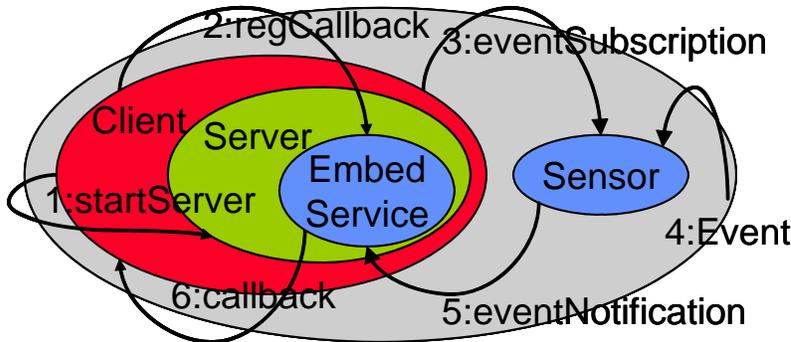


Figure 40 Client - Server Notification



Technical Report

A Taxonomy of Sensor Network Architectures

D.T. Fokum, V.S. Frost, P. Mani, G.J. Minden,
J.B. Evans, and S. Muralidharan

ITTC-FY2009-TR-41420-08

July 2008

Project Sponsor:
Oak Ridge National Laboratory

A Taxonomy of Sensor Network Architectures

D.T. Fokum^{a,*}, V.S. Frost^{a,*}, P. Mani^{a,2}, G.J. Minden^{a,2},
J.B. Evans^a, S. Muralidharan^{b,1,2}

^a*University of Kansas, Information and Telecommunication Technology Center,
Lawrence, Kansas 66049, USA*

^b*Cerner Corporation, Kansas City, Missouri USA*

Abstract

Several architectures have been proposed for sensor networks. However, there is a lack of an over-arching sensor network architecture. Here we present some of the issues associated with existing sensor network architectures. Next we present several sensor network architectures, including one suitable for a multi-owner environment, classifying these architectures in terms of function and compositional elements. We also highlight each architecture's key attributes in order to identify their commonalities. In making our arguments we refer to the concept of invariants, which are components of a system that cannot be changed without losing backward compatibility [1]. Our results show that while several sensor network architectures exist, each with different attributes, these architectures share several invariants.

Key words:

Sensor Networks; Taxonomy; Sensor network architecture; Invariant

1 Introduction

Sensor networks are an emerging application of advanced wireless networking and computer technology. Sensor networks typically consist of a set of small resource-constrained computers, called sensor nodes that collect data from their environments and then transmit that data on to a base station, or other central site. In general a wireless sensor node (WSN) would consist of a sensing device, e.g., an electronic nose, a temperature sensor or a motion detector, a small microprocessor, a radio and a limited energy source. It should be noted that when a sensor node is connected to just one sensor, the sensor node is sometimes called a sensor, which causes some confusion [2]. Base stations, unlike wireless sensor nodes, will generally have radios, but will have available more computing resources and a larger energy source. The base stations will generally aggregate information from the nodes and then pass them on to other computers for presentation [2].

Sensor networks have been identified as being key technology in monitoring and detecting threats. These systems face critical technical challenges in providing a security and management architecture in scenarios representative of a large class of applications. The design and architecture of sensor networks

* Corresponding author.

Email addresses: `fokumdt@ittc.ku.edu` (D.T. Fokum), `frost@ittc.ku.edu` (V.S. Frost), `mpradeep@ittc.ku.edu` (P. Mani), `gminden@ittc.ku.edu` (G.J. Minden), `evans@ittc.ku.edu` (J.B. Evans), `satyam@ittc.ku.edu` (S. Muralidharan).

¹ Present address: Cerner Corporation, Kansas City, MO USA

² This work was supported in part by Oak Ridge National Laboratory under award number 4000043403 as part of the ORNL-SensorNet Initiative.

has been studied in [3, 4] and [5], while deployment experiences are recorded in [5–10]. However, a taxonomy for sensor network architectures still needs to be defined. This paper makes some steps to address this deficiency; here we classify sensor network architectures in terms of function and compositional elements. In addition we show that these sensor network architectures all possess invariants [1], which are system elements that cannot be changed without losing backward compatibility.

The rest of this paper is laid out as follows: In section 2 we define the attributes used in our architecture comparison. Section 2 also lists some of the issues with existing sensor networks. Section 3 presents several sensor network architectures, and highlights their attributes and invariants. Included in section 3 is a discussion of a new sensor network architecture focused on a multi-owner environment. Section 4 summarizes the findings from Section 3. We conclude the paper in section 5.

2 Related Work and Context for Discussion

An architecture decomposes a system into component parts. Additionally an architecture may also define structures and functions (interfaces) to its components. At its lowest level an architecture may define protocols and state machines for communications [11].

A new method for designing and evaluating networking protocols and architectures is proposed in [1]. It states that all systems contain invariants, i.e., components that cannot be changed without losing backward compatibility; for example IP addresses are an invariant in the current Internet. Explicit

invariants result from deliberate decisions to limit the flexibility of a system, while implicit invariants are the unplanned result of deliberate design decisions. A set of invariants may be evaluated using the following questions:

- (1) Is the set complete?
- (2) Is the set independent?

With individual invariants we also have to ask these questions:

- (1) Does an invariant affect many components or just a few?
- (2) Does an invariant affect many aspects of an architecture or just a few?
- (3) Does an invariant affect hardware or just software?
- (4) Does an invariant have security or privacy implications?
- (5) Does an invariant have internal flexibility?

Evaluation on these characteristics should help us determine the quality of the given architecture [1].

Some of the attributes that we will be using to characterize sensor networks include whether or not the architecture is agent-based, delay-tolerant or fault-tolerant. Whether or not the architecture supports data fusion, Internet connectivity, location encoding, metadata communications, or has support for security mechanisms. Finally, we will also evaluate architectures to see if they are context-aware, based on standards, or have tiered architectures. We define each of these attributes below to give some context to this discussion.

Agent-Based Incorporate an agent (piece of software) that travels between the nodes of an architecture to perform some task autonomously, while fulfilling the goals of the program that dispatched the agent [12].

Delay-Tolerant Used in instances where an end-to-end path may not be as-

sumed to exist between two nodes. Delay-tolerant networks can be conceptually partitioned into two parts, with a gateway serving as a link between both parts. The gateway node is assumed to have significant storage capabilities so that data may be buffered when an end-to-end path does not exist, and transmitted when a path becomes available [13].

Fault-tolerant Fault tolerant architectures are those which have the ability to deal with system faults such that service failures do not result [14] and [15].

Data Fusion Architectures that support data fusion have certain intermediate nodes within the architecture that process data from several sensors into a more concise representation, which is then retransmitted to the sink [16].

Internet Connectivity Support This attribute is used to characterize architectures that contain a node, or several nodes, that can be used to bridge the connection between the global Internet and the sensor network. Justification and examples for this architecture may be found in [6, 13, 17].

Location Encoding Sensor networks supporting this attribute have the ability to store sensor readings with location information so that the two may be correlated.

Metadata communications In sensor networks that support this attribute data is read and stored on sensors, and the sensors forward messages (metadata) describing the data that was read to the sink. The metadata is then used to query the sensor network [18].

Security Support Sensor networks supporting this attribute should provide the following services: data confidentiality (data should not be leaked to unauthorized users), data authentication (proof that a message was actually sent by a given user), data integrity (proof that a given message is the same as that which was sent), and data freshness (ensures that data is recent,

and could not be a replayed copy of a message) [19].

Context awareness Context-aware sensor networks are cognizant of their environments. Applications running in context-aware sensor networks would typically have sensors for collecting context information, a set of rules on how to act given context, as well as a set of actuators to carry out actions [20].

Standards Based This attribute is used to describe sensor networks that are based on some standard, such as the Open Geospatial Consortium's (OGC) Sensor Web Enablement standards, or the IEEE 1451 standard. It is important to identify this attribute, as we shall see in the following sections that sensor networks have evolved largely free of any standardization.

Tiered architectures These architectures consist of different layers of sensor nodes (as in [5, 9, 10, 18, 21, 22]) or different layers of programs (as in [23]).

In this paper we will be classifying sensor networks in terms of the above attributes, and determining which of those attributes are invariants. However, others (References [24] and [25]) have suggested alternate approaches for evaluating sensor network architectures. In particular [24] states that sensor network architectures be evaluated in terms of the design objectives for sensor nodes. Reference [24] suggests the following architectural design attributes for sensor nodes: small physical size, low power consumption, concurrency-intensive operation (that is acquisition of sensor data, local processing of data followed by simultaneous transmission of data from several nodes to a base station), diversity in design and usage, robust operation, security and privacy, compatibility, and flexibility. Tilak *et al.* [25] suggest another approach for classifying sensor networks. They state that sensor networks may also be classified by communication models, data delivery models, and network

dynamics models. In making their arguments, Tilak *et al.* suggest that energy efficiency/system lifetime, latency, accuracy, fault-tolerance and scalability metrics be used to evaluate sensor network protocols. Next they state that sensor networks may be viewed in terms of infrastructure, network protocol and application/observer interests. Communication in a sensor network may be classified as either application or infrastructure. Application communications arise from informing the observer about sensed data. Application communications may be further characterized as either cooperative or non-cooperative. With cooperative communications, sensors cooperate with other sensors to fulfill the observer's need; non-cooperative sensors do not cooperate. Infrastructure communications on the other hand relate to the communications needed to configure, maintain and optimize the network. Sensor networks can be classified by application requirements for data delivery as continuous, event-driven, observer-initiated or hybrid. Sensor networks can also be classified in terms of network dynamics models. They may be classified either as static sensor networks or dynamic sensor networks. In this paper we do not classify sensor networks by any of these criteria. Instead we classify sensor networks by the attributes defined above, while indicating which of these attributes are invariants for the given architecture. In the next subsection we begin our examination by identifying some of the problems with current sensor network architectures.

2.1 Issues with Existing Architectures

The reader may conclude that only a limited number of sensor network architectures exist. In fact [26] distinguishes just two possible architectures for

wireless sensor and actuator networks — namely semi-automated and automated architectures. In semi-automated architectures a central base station coordinates the activities of the sensor nodes and the actuator nodes. In automated architectures a base station is not required, instead the actuators are programmed to operate and respond to events autonomously. Rather than limit ourselves only to two types of sensor network architectures, we contend that the number of sensor network architectures is much richer. In this paper we present a number of these architectures classified by function. Prior to presenting these architectures we argue that there is no overall sensor network architecture.

References [11] and [27] observe that sensor networking research is fragmented. In particular [11] argues that research into sensor networks is impeded by “the lack of an overall sensor network architecture” and not by any specific technical challenge. Moreover, it argues that while complex systems have been built by ignoring boundaries between subsystems, a sensor network architecture should be developed to allow others to extend previous work. This sensor network architecture will be akin to the architecture that has facilitated the growth of the Internet. The claim is that sensor networks will thrive if there is “a narrow waist in the architecture,” called the Sensor-net Protocol (SP) to allow protocols to evolve. SP will be a single hop protocol, but is analogous to IP. Below SP will be different link, MAC and physical layers, whereas above SP will be different sensor-application protocols. It should be noted that this sensor network architecture is slightly different from the OSI and Internet architectures since sensor networks mainly collect, aggregate and disseminate data, while the Internet is mainly concerned with end-to-end communication. One final requirement of the proposed sensor network architecture is that it

must allow cross-layer interactions between layers for more efficient sensor network operation [11].

Reference [27] also observes that sensor networking research is fragmented; however, it does not go as far as reference [11]. Instead it argues that better integration in sensor networks research may be achieved by using the following: a “hardware abstraction for new sensor node prototypes,” “abstract model of power consumption,” and a “protocol architecture scheme” for wireless sensor networks. The benefits of a protocol architecture include the following: it may facilitate the passing of packets between different layers of a protocol stack, and it may also help organize how information should be exchanged between different layers of the protocol stack [27].

From the discussion above it is evident that there is a lack of consensus on an over-arching sensor network architecture. The examples that we will present in section 3 will go towards highlighting the lack of an over-arching architecture. However, we argue that while there is a lack of an architecture, some similarities exist in sensor network architectures, in terms of their invariants and their functions.

3 Architecture Taxonomy

In this section we present a number of sensor network architectures and classify those architectures in terms of the attributes presented in Section 2. For each architecture we will also classify each of its attributes in terms of invariants, as introduced in Section 2.

3.1 *Architecture Classification*

In section 2 we discussed some issues with sensor networks and sensor network research. In this subsection we classify some successful sensor network architectures by decomposing each architecture into components.

Sensor networks have been successfully deployed to study birds on Grand Duck Island, Maine [5,9,10]. This sensor network used a multi-level architecture with sensor nodes performing computation and networking at its lowest level. The sensor nodes are grouped into a sensor patch, which is linked to a gateway node at the next level. The gateway transmits packets from the sensor patch to one or more base stations. These base stations provide database services as well as Internet connectivity. Finally, the last level consists of remote servers to support analysis, visualization and web content [5]. The reader may consult Fig. 2 in [5], Fig. 1 in [9], or Figure 1 in [10] for a system architecture diagram.

Reference [10] goes beyond the simple architecture presented in [5] to present an architecture that organizes all the sensor nodes within a sensor patch into a routing tree. In addition computation located within the sensor network so as to reduce the energy consumption of the individual nodes as well as reduce the volume of data being transmitted. Here the sensor network also has an independent verification network whose sole purpose is to generate independent data that can be used to corroborate readings from the sensor network. The verification network will consist of fewer, but more established sensor nodes. In addition to presenting the basic architecture discussed above, [10] also gives examples of sensor networks whose architectures are extensions of the basic architecture presented in [5,9,10]. One of these extensions uses Tiny Diffusion,

a routing protocol to establish communications between sources and sinks. With this architecture the network is aware of data naming and can apply filters. Another extension of the architecture uses the Tiny Application Sensor Kit (TASK) with a TinyDB database. With this architecture the sensor nodes have an SQL-variant query interpreter running on each node, and sensor nodes receive queries in an epidemic fashion [10]. The key attribute for this family of architectures is the tiered architecture. The Tiny Diffusion architecture has the additional attribute of supporting data diffusion, but we deal with this attribute at the end of the next paragraph. The tiered architecture may be seen as an explicit invariant since it results from a decision to limit the amount of processing that is done on the end nodes of this sensor network due to their limited computing power.

A slightly more complex architecture uses Directed Diffusion, which establishes *n-way* communications between one or more data sources and sinks [28]. The communications architecture is based on directed diffusion, matching rules and filters. Directed diffusion disseminates information in the distributed system, while matching rules identify when data has reached its destination. Finally filters process the data while it is en-route. This architecture can be seen as a method for performing in-network aggregation of data in a sensor network, thereby leading to a reduction of traffic in the sensor network [28]. The key attribute for this architecture is the support for data fusion, which is an implicit invariant since it results from the deliberate decision made to support communications between *n* sources and one sink. The task-awareness of sensor nodes is another attribute of this architecture, where task-awareness means sensor nodes store and interpret the data interests of other nodes.

Reference [29] can be viewed as an extension of Directed Diffusion [28]. It as-

sumes that nodes within a sensor network are named, and each node is within radio range of several nodes. Communication from the sensor network to the outside world is assumed to take place through some key nodes. Observations refer to readings from sensors, while certain collections of observations constitute an event, e.g., elephant-sighting event. Upon detection of an event data is sent to external storage for further processing. In addition data is stored by name within the sensor network, i.e., data-centric storage. Data-centric storage is preferable if the sensor network is large, i.e., contains many nodes, or if the sensor network detects many events, but not all the event types are queried. The data centric storage is supported by a geographic hash table (GHT), which provides a (key, value)-based memory. GHT uses Greedy Perimeter Stateless Routing (GPSR) for routing [29]. The key attribute for this scheme is the location encoding scheme. The implementation of the location encoding scheme, the GHT, constitutes an explicit invariant since the memory is deliberately limited to the (key, value) pair. Another invariant for this architecture is connectivity with the outside world through a limited set of nodes, which may be seen as an explicit invariant since the network is being deliberately limited.

Wireless sensor networks are typically composed of resource-limited nodes. As a result we need efficient algorithms to communicate in this environment. One suggestion is to use a data handling architecture that will support efficient spatio-temporal querying of data [30]. The design goals for this architecture include: multi-resolution data storage, distributed communication and computation load, and adaptability to correlations in sensor data. Temporal data reduction is only done at a single node, and has no communication overhead; once this data reduction is performed, only potentially interesting events are

reported to the rest of the sensor network. The DIMENSIONS architecture assumes a clustered sensor network with location encoding; as a result some of its attributes include a tiered architecture with location encoding support. The invariant for this architecture would be the implementation of the location encoding scheme.

A two-tier storage architecture (TSAR) for sensor networks, is yet another proposed sensor network architecture. With TSAR sensors transmit metadata rather than send actual sensor readings, since the metadata which may be a lot smaller than the actual data itself. Design of TSAR is based on the following principles: 1) Store locally, access globally; 2) Distinguish data from metadata; and 3) provide data-centric query support. At each proxy tier TSAR uses an Interval Skip Graph for storing data (The interval skip graph is an ordered, distributed structure that allows one to locate all intervals containing a particular value or range of values.) At the sensor level TSAR implements a local archival store and a mechanism to allow sensors to adapt to changing data and query characteristics. The TSAR scheme was field-tested, and experimental results show that TSAR displays good performance in a multi-tier sensor network [18]. The key attributes of this scheme are the metadata communications and the tiered architecture. The invariant for this architecture is limited to the Interval Skip Graph, which can be seen as an implicit invariant since the coarseness of the data intervals influences the resolution of query results. Related to this invariant is the adaptive summarization scheme, which allows sensor nodes to adjust the frequency of sending data updates with the probability of not being able to fulfill a query. Please consult Figure 1 in [18] for a logical view of the TSAR architecture.

Middleware can also be incorporated into a sensor network architecture. Römer

et al. [31] state that sensor network middleware should be geared to support the development, maintenance, deployment, and execution of sensing applications. In addition, they [31] state that sensor network middleware should possess the following attributes:

- It must provide ways of putting application knowledge into the sensor network,
- It should integrate communication and application-specific data processing closely,
- Provide ways to support automatic configuration and error handling.
- Support for time and location management.

Shen *et al.* [32] introduce middleware called Sensor Information and Networking Architecture (SINA). SINA allows sensor applications to issue queries and commands, collect query results and monitor the sensor network. The SINA architecture consists of hierarchical clustering — allows sensor nodes to aggregate into clusters, — attribute-based naming — which allows users to query the sensor network by some attribute, e.g., what is the average temperature in a given quadrant, — and location awareness, which requires sensor nodes to know their physical location, for example by using GPS. SINA [32] also provides the following attributes:

Information abstraction , that is the sensor network is conceptually seen as a collection of attributes of each sensor node.

Sensor Query and Tasking Language (SQTL) , which serves as an interface between sensor applications and the SINA middleware.

Sensor Execution Environment (SEE) , which runs on each sensor node and dispatches all incoming messages, examines all incoming SQTL mes-

sages, and performs the operations specified by each message. SEE also handles outgoing messages.

Built-in Declarative Query Language to give users the ability to submit a query directly instead of submitting an SQTL script.

Dyo [22] suggests middleware that can be used in sensor networks to support data retrieval applications with mobile data collectors. This paper observes that not very much research has been done on data collection using mobile sinks. Consequently the paper develops a scalable, energy-efficient, distributed spatial index that adapts to the sensor network query and data update rates. The proposed index uses a static clustering algorithm and proactive and reactive modes for index updates [22]. The key attributes of this architecture are the tiered network architecture and the distributed spatial index for querying of the network. The spatial index can be seen as an explicit invariant since it now requires all queries submitted to the sensor network to now contain information about the area of interest for the query.

Another application for sensor networks is to fuse data from several sources using a fusion application, and present the fused data to a user. A fusion application is continuous in nature, requires efficient transport of data from sources to sinks, and it also requires efficient in-network processing of application fusion functions. Ramachandran *et al.* [33] present a fusion architecture for sensor networks called DFuse in [33]. Informally, the DFuse architecture consists of the following: an application task graph — showing the data flows and relationships amongst the fusion functions, — code for the fusion functions, and a cost function that formalizes some metric for the sensor network. Note that the fusion functions may be placed anywhere in the sensor network, subject to the cost function being satisfied. In addition every node in the WSN

has a network layer that allows it to reach any node within the WSN [33]. More formally, the two main parts of the DFuse architecture are the Fusion Module and the Placement Module. The Fusion module performs the following tasks:

- Structure management (handles the channels used for fusion functions - fusion channels. This management includes migrating the channels to other nodes)
- Correlation control (handles specification and collection of data supplied to the fusion code)
- Computation management (handles specification, application and migration of fusion functions)
- Memory management (handles caching, prefetching and buffer management)
- Failure and latency handling (deals with sensor failure and communication latency. It also allows fusion functions to operate on partial data sets.)
- Status and feedback handling (handles interaction between data sources and fusion functions.)

The main responsibility of the Placement module is to create an overlay of the application task graph onto the physical network that best satisfies an application-defined cost function [33]. The key attribute of this architecture is support for data fusion, including the code that performs the fusion functions. Thus fusion support may be seen as an explicit invariant since it deliberately limits the user from getting fine-grained data from a sensor network. Related to this invariant, is the fusion channel, which is itself an explicit invariant. The fusion channel is an invariant since it provides interconnection between different parts of the system. For a diagram summarizing the DFuse architecture, please consult Fig. 2 in [33].

The last major class of sensor network architectures is based on databases. Yao and Gehrke [34] advocate a database approach to sensor networks, since declarative queries are suited for sensor networks. They propose using a query proxy on each sensor node that lies between the network layer and the application layer on that sensor node. Another reason for advocating the use of databases is that communication is more expensive than computation in sensor networks. Databases allow computation to be moved from nodes outside of the network to nodes within the network. With this approach, a query optimizer located on the sensor network's gateway node. The query optimizer generates a distributed query processing plan for queries generated from outside of the network. The query plan is sent to all nodes, and the gateway node responds to the query with the records coming back to the gateway node [34]. The key attribute for this architecture is the tiered network architecture. In particular the query proxy layer constitutes an invariant for this architecture. The query proxy layer may be viewed as an implicit invariant, since all queries are now required to be submitted to the query optimizer node in a network.

3.2 Standards-Based Sensor Networks

In the previous subsection we saw that many previous sensor networks have been marked as one-off designs generally devoid of any standardization. Recall from section 2.1 that there is no protocol akin to IP for sensor networks. Recently we have seen an emerging class of sensor networks that include open standards in their development, for example architectures based on the Open Geospatial Consortium (OGC) Sensor Web Enablement standard [35, 36].

Reference [37] makes the case for the use of standards in sensor networks, par-

ticularly those used for homeland security purposes. This paper states that open, standardized sensor interfaces and sensor data formats are needed to effectively integrate, access, fuse and use sensor-derived data for homeland security applications. The paper goes on to argue that without open, standardized interfaces and data encoding schemes it will be impossible to integrate a wide variety of sensors and networks. Open sensor interface standards such as the IEEE 1451 [38] and Universal Plug and Play (UPnP) [39] standards provide ways to interface transducers to networks. Meanwhile, Sensor Web Enablement (SWE) standards offer methods for sensor system discovery and control based on the Internet and the OGC's geo-processing framework. In summary, reference [37] states that the following standards are necessary for the development of a homeland security sensor network: transducer interface standards based on IEEE 1451 and web-based application interfaces. The key attributes of the sensor network proposed for homeland security include hardware-based fault tolerance [40], Internet connectivity support, location encoding, security support, and a standards-based architecture. Of these attributes location encoding and the standards-based architecture may be considered implicit invariants, since the location encoding scheme requires that data be stored with locations encoded in a specific format, while the standards-based architecture deliberately requires all sensor interfaces to comply with a given standard.

The OGC Sensor Web Enablement standard addresses the problem of having isolated, custom-designed sensor networks with incompatible sensor standards. Reference [36] introduces the sensor web enablement (SWE) specifications. These specifications include:

- Standard constructs for accessing and exchanging observations and measurements.

- Sensor Model Language (SensorML) Implementation, which provides an information model that enables the discovery and tasking of sensors.
- Transducer Markup Language (TML) Implementation, which provides a method for describing information about transducers.
- Sensor Observation Service (SOS) Implementation, which allows standard access to observations from sensors and sensor systems.
- Sensor Planning Service (SPS) Implementation, which specifies interfaces for a service to participate in collection feasibility plans.
- OpenGIS Sensor Alert Service, which allows users to subscribe to specific alerts, and determines the nature of offered alerts, and the protocols used for those alerts.
- OpenGIS Web Notification Service (WNS) Interface, which allows a client to have asynchronous communication with other services.
- A universal method for connecting transducer interfaces and application interfaces, such as the IEEE 1451 for smart transducers. The IEEE 1451 standard is an object-based protocol that allows sensors to be made accessible to clients over a network. The IEEE 1451 standard allows sensors to be accessible to clients across a network using Network Capable Application Processor (NCAP), which is the point of interface between the application and transducer interfaces.

An example of an architecture that uses the SWE standards is SensorNet [35]. This architecture uses standards from the OGC to learn the location of every sensor and measurement and help with interoperability. Interoperability is enhanced in this architecture by making use of web services for application interfaces. In particular this architecture uses the ORNL SensorNet node to host middleware that interfaces between the sensors and remote users and

applications. The ORNL SensorNet node is directly connected to the Internet, and it also hosts a web server to allow for intelligent processing, as well as any local processing of data. Another way in which this architecture tries to facilitate interoperability is by representing sensor data using “features,” which is an XML-like representation of data and sensor entities [35]. The key features of a sensor network based on the OGC Sensor Web Enablement standard are summarized in Figure 2 in [35].

The key attributes of the SensorNet architecture include fault-tolerance (each SensorNet node is equipped with two communication links for redundancy purposes), Internet connectivity support, location encoding, security support, a standards-based architecture that is also tiered. Of these attributes the location-encoding scheme is an implicit invariant, since locations must be encoded with a certain format. Recall that this architecture also has Internet connectivity support; therefore, extending the argument from section 2 we can also conclude that IP is an invariant for this architecture.

3.3 Internet-Connected Sensor Networks

While some sensor networks have possessed the ability to connect to the global Internet, in general, Internet connectivity support has not been a major consideration for sensor networks. One of the earliest references on sensor networks [41] argues for the use of multi-hop communications in sensor networks. These authors go on to state that work needs to be done to investigate how to link sensor networks to the global Internet. This statement is motivated by the fact that many current Internet protocols do not take the need to conserve energy very seriously. In addition this paper states that work needs to be

done on evaluating where processing and storage should take place in a sensor network [41]. For a logical view of this architecture, please consult Figure 2 in [41]. The attributes for this architecture include the tiered network architecture as well as support for conventional network services. The invariant in this architecture appears in the gateway that serves as the interface between the sensor network and the conventional network service. In general removal of any functionality from the gateway node will lead to a loss in backward compatibility of that node.

One example of a sensor network that has Internet connectivity is IrisNet (Internet-scale Resource-Intensive Sensor Network services), which aims to provide software components for a world-wide sensor network [6, 7]. These authors state that their sensor network is broader than the traditional definition of sensor networks, and includes Internet-connected, dispersed PC-class nodes. Such a sensor network must provide the following services:

- Planet-wide local data collection and storage.
- Real-time adaptation of collection and processing
- Data as a single queryable unit
- Support for queries posed anywhere on the Internet
- Data integrity and privacy
- Robustness
- Ease of service authorship

Under IrisNet service authors will have to figure out how to collect data, as well as how to query the collected data. IrisNet uses a two-tier architecture consisting of sensing agents and organizing agents. Sensing agents provide a generic data acquisition interface for sensors, while organizing agents collect

and organize data to respond to a query. Each sensor agent controls one or more senselets. Each senselet allows one to upload and control the execution of code in a sensor. As was the case with the OGC's Sensor Web Enablement standards, sensor-derived data is represented in XML in IrisNet. It should be noted that IrisNet has been deployed to monitor the Oregon coastline [6]. Please consult Figure 1 in [6] for an IrisNet architecture diagram, showing the organizing agents and the sensing agents.

The attributes for the IrisNet architecture include the agent-based architecture, Internet connectivity support, and the tiered network architecture. Moreover the invariants for this architecture include the agents, which may be seen as explicit invariants, and IP addresses, if we extend the example on invariants from section 2 to this architecture. The requirement to represent sensor-derived data in XML may also be seen as an explicit invariant.

Another architecture that allows access to sensor networks from Internet-type networks is the Janus architecture [17]. A prototype of the Janus architecture has been used to connect a sensor network with hosts on a network LAN. Janus uses an engine (This is a program running on the sink that provides an interface to sensor network functionality. The agent uses the engine to discover resources and functionality provided by the sensor network) running on the sensor network's sink as well as an agent that communicates with the engine. The engine and the agent communicate using eXtensible Resolution Protocol (XRP). The agent and engine exchange XRP messages to:

- Discover which sensor network resources are available;
- Send queries from the agent to the sink node on sensor network state; and
- Send information from the sink to the agent concerning the sensor network

state.

Janus also supports multiple access applications for sensor networks. All XRP messages are transported between the agent and the sink node using UDP. The use of XRP allows for expressive messages – that is XRP queries may be interpreted – to be exchanged between agent and sink. Use of XRP also allows for modularity in network design [17]. Figure 2 in [17] shows the extended architecture for sensor networks that use Janus for Internet access.

The attributes for Janus include the agent-based architecture as well as the support for Internet connectivity. The invariants for this architecture comprises of the agent and the engine. These components are considered invariants since they enable communications between the sensor network and the Internet. Adding functionality to one of these components without adding to the other will result in the loss of backward compatibility.

By no means do we claim that the two examples of Internet-connected sensor networks constitute an exhaustive list. More recently a group of researchers has come formed an Internet Engineering Task Force (IETF) working group to study routing over low-power and lossy networks, such as sensor networks [42].

3.4 Context-Aware Sensor Networks

A novel class of sensor networks is the group of context-aware sensor networks. Incorporating context into a network can have implications for energy efficiency. For example suppose sensors are equipped with light sensors, and it is known that temperature changes less frequently after dark. Sensors in this sensor network can then wake up and make temperature readings less

frequently once night falls.

An argument for building a context aware sensor network is presented in [43]. Reference [43] argues that if each sensor node is context-aware, then the entire network will be context-aware. In making this argument, these authors assume that a context-aware sensor network (CASN) is node-centric. They state that the goal of designing a context-aware sensor network is to prolong the life of the network. The CASN is composed of middleware running on sensor nodes. The middleware is composed of the following components: context representation (CRP), context interpretation (CI), context aware services (CAS), and a sensor society kernel (SSK). The CRP provides context availability, the CI interprets the context, the CAS manages services, and the SSK allows each sensor node to act as a member of a larger society — the sensor network. Finally, these authors suggest using a role-based local storage scheme (RBLS) to store contexts on a sensor node [43]. Figure 2 in [43] summarizes the architecture for a context-aware sensor network.

The key attribute for this architecture is context awareness. The middleware on sensor nodes also hints to the fact that this architecture is tiered or layered. The invariant in this architecture is the middleware. If any of its components is changed, we can lose backward compatibility. For example if the context interpretation module is changed it may return states that the other middleware components do not know how to handle.

3.5 *Agent-Based Sensor Networks*

Agents can also be used in designing sensor networks, as we have already seen from [6]. The use of agents in a sensor network architecture allows for flexibility in that architecture since the sensor network can be quickly reprogrammed to perform a different task.

Reference [12] makes the case for the use of mobile agents in sensor networks. This paper observes that sensor networks are moving towards a single deployment, multiple applications paradigm; however, sensor nodes may not necessarily have the capability to store all the programs needed for the different applications. Mobile agents can be used as an option for dynamically deploying applications to sensor networks. Some examples of use might include:

- Deploying mobile agents to a visual sensor network to collect reduced data from some region of the WSN and query the data set for some information.
- Using mobile agents for target tracking and object recognition in a sensor network.

According to [12], two types of sensor networks — hierarchical or flat — may be distinguished. In hierarchical (clustered) sensor networks mobile agents may be either deployed by a cluster head to visit all nodes within the cluster, known as the intra-cluster method, or they may be deployed by the sink node of a sensor network to visit all the cluster heads, known as the inter-cluster method. In flat sensor networks the sink node can dispatch a “mother agent,” which visits a target region of the sensor networks. Once in the sensor network the mother agent will dispatch child agents to visit the nodes in the target region and collect information that will either be carried directly to the sink

or to the mother agent.

It should be noted that mobile agents are frequently implemented in middleware. This middleware may be either coarse-grained or fine-grained, where coarse-grained agents typically have smaller code sizes with lower re-tasking flexibility while fine-grained agents have larger code sizes with higher re-tasking capability. In addition, having multiple agents cooperate can actually lead to an improvement in performance of the entire WSN [12]. Figure 4 in [12] summarizes the architecture of one type of mobile agent-based sensor network architecture. Note that the battleship in the figure represents the mother agent deployed by the sink node, and the arrows represent data flow to and from the mother agent.

The agent-based architecture and support for data fusion, particularly in sensor networks with mobile agents, form the set of attributes for this family of sensor network architectures. The invariants for this architecture are the agents.

3.6 Service-Oriented Sensor Networks

An emerging trend in sensor network architectures is the deployment of service-oriented sensor network architectures. Architectures such as these permit the incorporation of a diverse set of platforms and allow sensor nodes to discover the capabilities of other nodes by querying a service repository.

Rezgui and Eltoweissy [44] introduce a service-oriented architecture for sensor-actuator networks, called SOSANET. In proposing their architecture Rezgui and Eltoweissy [44] argue that existing sensor network architectures are application-

specific. Service-oriented network architectures can address this issue by allowing future sensor network designers to pick components from different sensor networks and integrate these into a new sensor network application. Sensor-actuator networks (SANETs) are different from ordinary sensor networks in that they include actuators that are able to change the environment of a sensor network. One example of a SANET can include a sensor network that has heat sensors and fire sprinklers. If the heat sensors detect combustion, the sensors will notify the sink, and the sprinklers can be triggered to douse the flames. SANETs may be classified as either generic or customizable. Service-oriented sensor-actuator networks are a type of customizable SANET.

Each node in a SOSANET exposes its capabilities as services. Each node in the SOSANET has a service directory showing the capabilities provided by reachable nodes. The service directories are used to perform service-driven routing in the SOSANET. Users get information from the SOSANET by submitting queries to either the base station or one of the nodes in the SOSANET. The queries may be either classified as task queries or event queries. Queries specify an event, condition, action, spatial scope, and temporal scope (ECAST) when invoked [44].

The architecture for the SOSANET consists of a service-oriented query (SOQ) layer, which receives queries from the service-driven routing layer, interprets them, invokes the services necessary for the query, collects the service results, packages the services' results into query results, and submits the query results to the query issuer. This layer consists of a service invocation scheduling module and an event detection module. When a query is received at a node it is submitted to the event detection module, which checks for the existence of a given condition. When the condition is detected, the query is submitted

to the service invocation scheduling module. Above the service-oriented query layer is the service layer, which contains the implementation of all services in the SOSANET. The architecture also includes a routing layer that delivers queries to the SOQ layer, sends out query results from the SOQ layer, and forwards received queries and query results. The routing layer is composed of the service-driven routing protocol (SDRP) and the trust-aware routing protocol (TARP). The former routes queries from the base station to sensor network nodes, while the latter forwards results from sensor network nodes back to the base station [44].

It should be noted that the proposed architecture has been implemented in TinySOA. Simulation results show that SDRP is an energy-efficient routing protocol. TinySOA is also shown to be more energy efficient than TinyDB. In addition TinySOA queries have a shorter response time than TinyDB queries. Finally SANETs based on TinySOA can be more deployed more rapidly than sensor networks based on TinyDB, since queries are automatically discovered under TinySOA. Figure 1 in [44] summarizes the key components of a SOSANET node.

The attributes for SOSANET include the service-oriented architecture as well as the layered, or tiered, architecture. The invariant for this architecture is the service-oriented query layer, which performs a lot of the processing necessary to receive query results. Improper changes to this layer can result, for example, in the query issuer not being able to interpret the query results.

Another service-oriented sensor network architecture is found in [45]. This three-tiered service-oriented sensor network architecture has been used to integrate with RFID and monitor hazardous chemicals for a petroleum company.

The tiered architecture allows sensor nodes with a range of capabilities to be integrated into a large-scale sensor network. The layers of the architecture consist of the backend, gateway, and front-end. The back-end (application) layer consists of the following:

- Service repository, which contains a database of all services available in the sensor network,
- System state manager, which keeps track of the states of the sensor nodes
- Service mapper, which maps the services to different nodes
- Service invocation manager, which contacts all the nodes running a given service and returns the results of that service invocation to the application, and the
- Notification manager, which uses a web service to distribute event messages.

The gateway (platform abstraction) layer facilitates interoperability between sensor platforms. In particular this architecture uses Universal Plug and Play (UPnP) [39] as the interface between the application layer and the sensor network. The gateway layer performs the following functions: message transformation — translating between packet-level proprietary sensor network messages and UPnP arguments, — and assisting in the deployment of services to the sensor network. One key feature of the gateway layer is the dynamic instantiation of service proxies. The service proxies — which are virtual representations of the service interfaces — are instantiated whenever a service is provided by the sensor network and destroyed whenever the service becomes unavailable.

The front-end (device) layer incorporates the multitude of sensor networking and RFID devices. Some of the functions provided by this layer include:

- Reliable dissemination of messages to nodes — this allows new service exe-

cutables to be transferred reliably to nodes

- Platform-dependent service executables
- Event detection and alarms — this allows timely detection and reporting of special conditions to a central node, – and platform-specific networking protocols.

This architecture was successfully deployed in a trial with an oil company, and the architecture was shown to be feasible; however, more work needs to be done to make the architecture more scalable. Figure 3 from [45] summarizes the key features of this architecture.

The attributes for this architecture include the tiered architecture, which is also service-oriented. On the other hand the invariants for this architecture include the gateway layer and the service repository layer. For example, improper changes in the service repository layer can prevent other nodes from knowing the locations of other services, while changes in the gateway layer can prevent the correct translation of network messages and UPnP arguments.

3.7 Secure and Fault-Tolerant Sensor Networks

Another emerging trend is sensor networks that include security and fault-tolerance from the time of design [46] and [47]. No architecture is presented in [47], but this paper presents a scheme for enhancing the reliability of sensor networks. When a sink has little energy left, the sink is relocated to another sensor node [47].

Reference [46] presents an architecture for a secure and survivable wireless sensor network with heterogeneous nodes. The architecture will provide security

and survivability mechanisms and techniques, and security and survivability requirements and services. Since sensor networking applications need to be able to run continuously and reliably without interruption, survivability needs to be factored into the development of a WSN. The security requirements for a sensor network are: confidentiality, authentication, integrity, and secure management, while the survivability requirements include: reliability, availability, and energy efficiency.

The reader is referred to [46] to obtain more details on the architecture. It should be noted that [46] provides simulation results to show that if a small number of powerful sensor nodes have reasonable storage, processing and transmission capabilities when using the proposed scheme, then the WSN can have good key connectivity, reliability and resilience. In addition the simulation results show that there is a trade off between security and survivability in some scenarios. Figure 1 from [46] summarizes the key components of the secure and survivable sensor network architecture.

The key attribute for the secure and survivable sensor network architecture is security support. On the other hand the invariant here is the key management scheme. If a new key management scheme is chosen for a set of sensor nodes these sensor nodes can lose the ability to communicate with other sensor nodes.

3.8 Vehicle-Based Sensor Networks

In the near future, we will begin to see sensor networks deployed to vehicles to enhance driver safety, and allow drivers to pick the best route between two points based on road conditions. Reference [48] describes an architecture for a

vehicular ad hoc network that is safety-oriented. In this architecture vehicles and roadside entities are seen as peers. The peers are organized in zones called peer spaces, while nodes in a peer space share a common interest. Peers may be organized into either cluster-based or peer-centered structures [48].

One protocol for vehicular ad hoc networks is Vehicular Information Transfer Protocol (VITP), which is an application-layer, stateless communication protocol analogous to HTTP [21]. The VITP architecture (infrastructure) consists of VITP peers (software components running on vehicle computers), a location encoding scheme, and additional protocol features for performance optimization, quality assurance and privacy protection. VITP stores location information as two-value tuples. When a vehicle needs some information, it formulates a query and broadcasts it. The dynamic collection of VITP peers that responds to a query is called a virtual ad hoc server (VAHS), in other words the VAHS is based on a query and target-location area. Simulation results for VITP performance show that the Return Condition for VITP requests is very important, since it affects the dropping rate of VITP transactions as well as the accuracy of VITP query results [21]. Figure 3 from [21] shows how protocols are layered in VITP.

The attributes for VITP include the tiered architecture and VITP's support for location encoding. The latter may be seen as an implicit invariant since all nodes must now use the same format for representing location information.

Another example of a mobile sensor network is found in [8]. This paper discusses a mobile sensor network composed of CarTel nodes that processes heterogeneous data. In general mobile sensor networks allow one to cover a larger surface area with fewer sensors. Each CarTel node consists of a mobile, embed-

ded computer connected to several sensors. The node runs software that functions as a gateway between the node and the rest of the sensor network. The architecture consists of a portal, which hosts CarTel applications and serves as a sink for data sent from the mobile nodes. There is also an ICEDB (Intermittently connected database), which is a delay-tolerant query processor. Finally, there is a CafNet (Carry and forward network), which is a delay-tolerant network stack. Unlike TCP, CafNet uses a message-oriented data transmission and reception API. This allows CafNet to be used in delay-tolerant networks. CafNet informs the sensor network applications when network connectivity is available, then the application decides which sensor network information needs to be sent. The CafNet communication stack consists of a Transport Layer, a Network layer and a Mule Adaptation Layer. The CafNet network layer supports buffering of some data [8]. Figure 2 in [8] shows the software architecture for CarTel.

The main attributes for CarTel include the delay-tolerant network architecture as well as the location encoding scheme. The invariant in this architecture is CafNet, the delay-tolerant network stack, which must continue to expose the same interfaces and services after any changes if backward compatibility is to be maintained.

Reference [49] discusses a network in which cars communicate with each other using TrafficView nodes to exchange data on the state of the road. According to [49] this form of inter-vehicle communication is different from traditional MANETs because of rapid changes in link topology, a frequently disconnected network, data compression/aggregation, prediction of vehicle's positions, and energy consumption not being an issue. A TrafficView node consists of the following modules: a GPS/OBD module, a receive module, a validation module,

an aggregation module, a send module, and a display module. Two algorithms that may be used for aggregating data (cost-based and ratio-based) in a vehicular network are discussed and evaluated in [49]. The results indicate that ratio-based aggregation works well in actual test conditions [49]. The components of a node architecture for TrafficView are found in Figure 4 in [49].

TrafficView’s set of key attributes includes location encoding and data fusion support. As was the case for VITP, the location encoding scheme can be seen as an implicit invariant since all nodes must now use the same format for representing location information.

The last class of vehicle-based sensor networks uses a system of train-based sensors to monitor wheel bearing temperatures [50]. This sensor network uses IEEE 802.11b for inter-car train communications, GPS information to provide location information. Backhaul communications are provided by a 1xRTT radio, and the train data is uploaded to a web server. Beyond the system specifications provided above, the architectural details of this sensor network are not available.

3.9 Habitat Monitoring Sensor Networks

As we observed in section 3.1, some of the earliest sensor networks were used for monitoring seabirds on Grand Duck Island in Maine [5], [9] and [10]. Reference [9] indicates that a tiered architecture was developed for this monitoring. At its lowest level are sensor nodes, which collect environmental data. The next tier consists of a sensor network gateway, which communicates with the sensor network and the transit network. At the next tier is the “remote base station

that provides WAN connectivity and data logging.” In order to provide some degree of fault-tolerance, each tier of the sensor architecture provides persistent data storage to guard against data loss. The architecture also provides data management services ranging from simple data logging to a full-fledged relational database service running on the base station. It is worth noting that the habitat monitoring sensor network also includes iPacs (known as gizmo in the paper) to allow for remote management of the sensor network [9].

3.10 Multi-owner Sensor Network Architecture (MOSN)

There is growing literature concerning the architecture and design of sensor networks [2–4], as well as the Open Geospatial Consortium Sensor Web Enablement efforts [51] and Oak Ridge National Laboratory’s SensorNet Information Architecture [52]. Several of these types of sensor networks have already been deployed [5].

A premise of this discussion is that elements of the sensor network will be owned by multiple organizations and communicate across administrative domains. Thus, there is a need for mechanisms that facilitate access to and control of sensors across multiple organizations as well as a requirement for rapid deployment. Ownership by a wide variety of administrative domains is briefly mentioned in [53]. While SensorML [35] has sensor schemas that include security, user limitations and access constraints (like `documentConstrainedBy`), and schemas that identify the responsible party (like `operatedBy`), the integration of these into an overall system remains to be explored.

The MOSN architecture extends ORNL’s SensorNet Information Architec-

ture and has been built upon the existing sensor network architectures (e.g., [6, 51, 53, 54]), to create a system based on the above concepts that facilitate the participation of multiple organizations in supplying needed component/subsystem functionality. A model of MOSN has been implemented and evaluated.

The objective of the MOSN is to develop a unified architecture that has elements owned/controlled by a variety of organizations which can communicate across administrative domains. The MOSN architecture is general, scalable (in size and evolution of technologies), flexible (able to mix and match technologies based on the venue requirements), economical (based on COTS technologies), and leverages standards where possible. The MOSN approach facilitates multiple organizations providing different services, enabling the development of a business model based on sensor network technologies.

MOSN components are divided into three layers, as shown in Fig. 1 in [23]. These layers include the following:

- The device layer, which is composed of all the sensor nodes, as well as the data access and management endpoints for the entire architecture.
- The repository layer, which forms a link between the lower device and the upper application layers to allow for information dissemination. This layer is composed of databases that store sensor data as well as databases that store information needed to support the system.
- The application layer, which presents a unified view of the different components of the architecture to the user.

Communication between the layers in the MOSN architecture is done by extending the Ambient Computing Environments (ACE) architecture [52, 55].

The device control and data flow mechanisms developed for ACE are used to manage connections between applications and sensor nodes. The ACE control mechanisms allow devices to be authenticated by a controlling application. In addition ACE allows access and control of devices to be based on an established security policy. Finally, the ACE data flow mechanism supports real time exchange of data between applications and devices that is private and checked for integrity. ACE supports establishing services within the environment to archive data flows, replicate data flows to multiple receivers, and play back archived data.

We conclude this section by observing that the key attributes of the multi-owner sensor network architecture include: Internet connectivity support, security support, a standards-based architecture that is also tiered and service-oriented. On the other hand the invariants include the service-oriented architecture, and the standards-based architecture. Due to Internet connectivity support, IP addresses may also be seen as an invariant for this architecture.

4 Architecture Comparison Summary

In the previous section we presented the key elements of different classes of sensor networks, including a new sensor network architecture suitable for a multi-owner environment. Those architectures were compared in terms of certain attributes. Table 1 summarizes the key features, while Table 2 summarizes the invariants of the architectures presented in Section 3.

	Agent-based	Delay-Tolerant	Fault-Tolerant	Fusion Support	Context Aware	Internet Connectivity Support	Location Encoding	Metadata Comms.	Service-Oriented	Security Support	Standard based	Tiered
Habitat monitoring system [5, 9, 10]						x						x
Directed Diffusion [28]				x								
Data-centric Storage [29]							x					

continued on next page

<i>continued from previous page</i>												
	Agent-based	Delay-Tolerant	Fault-Tolerant	Fusion Support	Context Aware	Internet Connectivity Support	Location Encoding	Metadata Comms.	Service-Oriented	Security Support	Standard based	Tiered
TSAR [18]								x				x
Middleware Design for Integration of Sensor Networks and Mobile Devices [22]												x

continued on next page

<i>continued from previous page</i>												
	Agent-based	Delay-Tolerant	Fault-Tolerant	Fusion Support	Context Aware	Internet Connectivity Support	Location Encoding	Metadata Comms.	Service-Oriented	Security Support	Standard-based	Tiered
DIMENSIONS [30]							x					x
DFuse [33]				x								
Cougar [34]				x								x
ORNL SensorNet [35,37]			x			x	x			x	x	x
WINS [41]						x						x

continued on next page

<i>continued from previous page</i>												
	Agent-based	Delay-Tolerant	Fault-Tolerant	Fusion Support	Context Aware	Internet Connectivity Support	Location Encoding	Metadata Comms.	Service-Oriented	Security Support	Standard based	Tiered
IrisNet [6, 7]	x					x						x
Janus [17]	x					x						
CASN [43]					x							x
MADSN and MAWSN [12]	x			x								
SOSANET [44]									x			

continued on next page

<i>continued from previous page</i>												
	Agent-based	Delay-Tolerant	Fault-Tolerant	Fusion Support	Context Aware	Internet Connectivity Support	Location Encoding	Metadata Comms.	Service-Oriented	Security Support	Standard-based	Tiered
Multiplatform Wireless Sensor Network work [45]									x			x
Secure and Survivable Wireless Sensor Networks [45]										x		

continued on next page

<i>continued from previous page</i>												
	Agent-based	Delay-Tolerant	Fault-Tolerant	Fusion Support	Context Aware	Internet Connectivity Support	Location Encoding	Metadata Comms.	Service-Oriented	Security Support	Standard-based	Tiered
VITP [21]							x					x
CarTel [8]		x					x					
TrafficView [49]							x					
Multi-owner sensor network architecture [23]						x			x	x	x	x

continued on next page

<i>continued from previous page</i>												
	Agent-based	Delay-Tolerant	Fault-Tolerant	Fusion Support	Context Aware	Internet Connectivity Support	Location Encoding	Metadata Comms.	Service-Oriented	Security Support	Standard-based	Tiered

Table 1. Summary of Architecture Features

Architecture Classification	Invariants	
	Explicit	Implicit
Habitat monitoring system [5, 9, 10]	Tiered architecture	IP
Directed Diffusion [28]		Data fusion
Data-centric Storage [29]	Location encoding scheme (GHT), Outside world connectivity support	
TSAR [18]		Interval skip graph, including adaptive summarization scheme
Middleware Design for Integration of Sensor Networks and Mobile Devices [22]	Distributed spatial index	
<i>continued on next page</i>		

<i>continued from previous page</i>		
Architecture Classification	Invariants	
	Explicit	Implicit
DIMENSIONS [30]		Location encoding scheme
DFuse [33]	Data fusion support, including fusion channels	
Cougar [34]		Query proxy layer
ORNL SensorNet [35,37]		Location encoding scheme, IP
WINS [41]		Interface code in WINS gateway, IP
IrisNet [6,7]	Agents, XML representation of data	IP
Janus [17]	XRP agent and XRP engine	IP
CASN [43]	CASN middleware	
MADSN and MAWSN [12]	Agents	
SOSANET [44]		Service-oriented query layer
46 <i>continued on next page</i>		

<i>continued from previous page</i>		
Architecture Classification	Invariants	
	Explicit	Implicit
Multiplatform Wireless Sensor Network [45]		Gateway layer and service repository layer
Secure and Survivable Wireless Sensor Networks [45]		Key management scheme
VITP [21]		Location encoding scheme
CarTel [8]	CafNet	
TrafficView [49]		Location encoding scheme
Multi-owner sensor network architecture [23]		Service-oriented architecture, standards-based architecture, and IP

Table 2. Summary of Architecture Invariants

5 Conclusion

In this paper we have presented a discussion of several sensor networks. From our discussion we have seen that there is no over-arching sensor network ar-

chitecture, as was previously argued in [11, 27]. However, from our review of sensor network architectures, we see that sensor networks share many features. In addition by examining their invariants (where invariants are components that cannot be changed without losing backward compatibility [1]) we also see that many architectures have several invariants in common, even if they are quite different.

Another contribution of this paper has been a discussion of an architecture, suitable for a multi-owner sensor network, developed at the University of Kansas. Unlike many of the other architectures presented in this paper, this architecture is not limited to low-powered sensor nodes, and in fact it has been used in conjunction with devices such as motes, Sun SPOTs, gumstix computers, and full-fledged PCs. However, it lacks certain features that some of the other architectures possessed, such as delay tolerance and context-awareness.

Sensor networks are increasingly being used to instrument our world. However, there is no single sensor network architecture, as one might find for the Internet. As was argued in [11] we conclude that sensor networks would be better able to fulfill their purpose if there is a single over-arching architecture. Some suggestions for developing such an architecture would be to identify and build physical, MAC, link and network layer protocols suitable for sensor networks. Above these layers we can build location-encoding schemes or any other applications or functionality needed by sensor network designers. Such a design might allow better portability of code and ideas from one sensor network to the next.

References

- [1] B. Ahlgren, M. Brunner, L. Eggert, R. Hancock, S. Schmid, Invariants: a New Design Methodology for Network Architectures, in: FDNA '04: Proceedings of the ACM SIGCOMM Workshop on Future Directions in Network Architecture, ACM Press, New York, NY, USA, 2004, pp. 65–70.
- [2] F. Zhao, L. Guibas, Wireless Sensor Networks: An Information Processing Approach, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004.
- [3] D. Estrin, D. Culler, K. Pister, G. Sukhatme, Connecting the Physical World with Pervasive Networks, *IEEE Pervasive Computing* 1 (1) (2002) 59–69.
- [4] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, Wireless Sensor Networks: a Survey, *Computer Networks* 38 (4) (2002) 393–422.
- [5] R. Szewczyk, J. Polastre, A. Mainwaring, D. Culler, Lessons From a Sensor Network Expedition, in: EWSN 2004: Proceedings of the First European Workshop on Sensor Networks, 2004.
- [6] P. B. Gibbons, B. Karp, Y. Ke, S. Nath, S. Seshan, IrisNet: an Architecture for a Worldwide Sensor Web, *IEEE Pervasive Computing* 2 (4) (2003) 22–33.
- [7] J. Campbell, P. Gibbons, S. Nath, P. Pillai, S. Seshan, R. Sukthankar, IrisNet: an Internet-scale Architecture for Multimedia Sensors, in: MULTIMEDIA '05: Proceedings of the 13th Annual ACM International Conference on Multimedia, ACM, New York, NY, USA, 2005, pp. 81–88.
- [8] B. Hull, V. Bychkovsky, Y. Zhang, K. Chen, M. Goraczko, A. Miu, E. Shih, H. Balakrishnan, S. Madden, CarTel: a Distributed Mobile Sensor Computing System, in: SenSys '06: Proceedings of the 4th International Conference on

Embedded Networked Sensor Systems, ACM Press, New York, NY, USA, 2006, pp. 125–138.

- [9] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, J. Anderson, Wireless Sensor Networks for Habitat Monitoring, in: WSNA '02: Proceedings of the 1st ACM international workshop on Wireless Sensor Networks and Applications, ACM Press, New York, NY, USA, 2002, pp. 88–97.
- [10] R. Szewczyk, E. Osterweil, J. Polastre, M. Hamilton, A. Mainwaring, D. Estrin, Habitat Monitoring with Sensor Networks, *Commun. ACM* 47 (6) (2004) 34–40.
- [11] D. Culler, P. Dutta, C. T. Ee, R. Fonseca, J. Hui, P. Levis, J. Polastre, S. Shenker, I. Stoica, G. Tolle, J. Zhao, Towards a Sensor Network Architecture: Lowering the Waistline, in: HOTOS'05: Proceedings of the 10th conference on Hot Topics in Operating Systems, USENIX Association, USENIX Association, Berkeley, CA, USA, 2005, pp. 24–30.
- [12] M. Chen, S. Gonzalez, V. C. M. Leung, Applications and Design Issues for Mobile Agents in Wireless Sensor Networks, *IEEE Wireless Communications* [see also *IEEE Personal Communications*] 14 (6) (2007) 20–26.
- [13] K. Fall, A delay-tolerant network architecture for challenged internets, in: SIGCOMM '03: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications, ACM, New York, NY, USA, 2003, pp. 27–34.
- [14] D. Hutchison, J. P. G. Sterbenz, Resilinet architecture definitions, Wiki (Feb. 6 2007).
URL <http://wiki.ittc.ku.edu/resilinet.wiki/index.php/Definitions>
- [15] R. J. Abbott, Resourceful systems for fault tolerance, reliability, and safety, *ACM Comput. Surv.* 22 (1) (1990) 35–68.

- [16] W. Chen, J. C. Hou, Handbook of Sensor Networks: Algorithms and Architectures, John Wiley & Sons, 2005, Ch. Data Gathering and Fusion in Sensor Networks, p. 495.
- [17] A. Dunkels, R. Gold, S. A. Marti, A. Pears, M. Uddenfeldt, Janus: an Architecture for Flexible Access to Sensor Networks, in: DIN '05: Proceedings of the 1st ACM Workshop on Dynamic Interconnection of Networks, ACM Press, New York, NY, USA, 2005, pp. 48–52.
- [18] P. Desnoyers, D. Ganesan, P. Shenoy, TSAR: a Two Tier Sensor Storage Architecture Using Interval Skip Graphs, in: SenSys '05: Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems, ACM Press, New York, NY, USA, 2005, pp. 39–50.
- [19] A. Perrig, R. Szewczyk, V. Wen, D. Culler, J. D. Tygar, SPINS: Security Protocols for Sensor Networks, in: MobiCom '01: Proceedings of the 7th Annual International Conference on Mobile Computing and Networking, ACM, New York, NY, USA, 2001, pp. 189–199.
- [20] G. Biegel, V. Cahill, A Framework for Developing Mobile, Context-aware Applications, in: PerCom 2004: Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications, IEEE Computer Society, 2004, pp. 361–365.
- [21] M. D. Dikaiakos, S. Iqbal, T. Nadeem, L. Iftode, VITP: an Information Transfer Protocol for Vehicular Computing, in: VANET '05: Proceedings of the 2nd ACM International Workshop on Vehicular Ad Hoc Networks, ACM Press, New York, NY, USA, 2005, pp. 30–39.
- [22] V. Dyo, Middleware Design for Integration of Sensor Network and Mobile Devices, in: DSM '05: Proceedings of the 2nd International Doctoral Symposium on Middleware, ACM Press, New York, NY, USA, 2005, pp. 1–5.

- [23] P. Mani, S. Muralidharan, V. Frost, G. Minden, D. Petr, A Unified Architecture for Sensor Networks with Multiple Owners, in: ACM SenSys 2008, Submitted.
- [24] J. Feng, F. Koushanfar, M. Potkonjak, System-Architectures for Sensor Networks Issues, Alternatives, and Directions, in: ICCD'02: IEEE International Conference on Computer Design, IEEE, IEEE Computer Society, Los Alamitos, CA, USA, 2002, pp. 226–231.
- [25] S. Tilak, N. B. Abu-Ghazaleh, W. Heinzelman, A Taxonomy of Wireless Micro-sensor Network Models, SIGMOBILE Mobile Computing and Communications Review 6 (2) (2002) 28–36.
- [26] F. Martincic, L. Schwiebert, Handbook of Sensor Networks: Algorithms and Architectures, John Wiley & Sons, Hoboken, NJ, 2005, Ch. Introduction to Wireless Sensor Networking, pp. 25–26.
- [27] V. Handziski, A. Kopke, H. Karl, A. Wolisz, A Common Wireless Sensor Network Architecture, in: Proc. 1. GI/ITG Fachgesprach "Sensornetze" (Technical Report TKN-03-012 of the Telecommunications Networks Group, Technische Universitat Berlin), Technische Universitat Berlin, Berlin, Germany, 2003, pp. 10–17.
- [28] J. Heidemann, F. Silva, C. Intanagonwiwat, R. Govindan, D. Estrin, D. Ganesan, Building Efficient Wireless Sensor Networks with Low-level Naming, in: SOSP '01: Proceedings of the 18th ACM Symposium on Operating Systems Principles, ACM, New York, NY, USA, 2001, pp. 146–159.
- [29] S. Shenker, S. Ratnasamy, B. Karp, R. Govindan, D. Estrin, Data-centric Storage in Sensornets, SIGCOMM Computer Communications Review 33 (1) (2003) 137–142.
- [30] D. Ganesan, D. Estrin, J. Heidemann, DIMENSIONS: Why Do we Need a New Data Handling Architecture for Sensor Networks?, SIGCOMM Computer

Communication Review 33 (1) (2003) 143–148.

- [31] K. Römer, O. Kasten, F. Mattern, Middleware Challenges for Wireless Sensor Networks, SIGMOBILE Mob. Comput. Commun. Rev. 6 (4) (2002) 59–61.
- [32] C.-C. Shen, C. Srisathapornphat, C. Jaikaeo, Sensor Information Networking Architecture and Applications, IEEE Personal Communications, [see also IEEE Wireless Communications] 8 (4) (2001) 52–59.
- [33] U. Ramachandran, R. Kumar, M. Wolenetz, B. Cooper, B. Agarwalla, J. Shin, P. Hutto, A. Paul, Dynamic Data Fusion for Future Sensor Networks, ACM Transactions on Sensor Networks (TOSN) 2 (3) (2006) 404–443.
- [34] Y. Yao, J. Gehrke, The Cougar Approach to in-network Query Processing in Sensor Networks, SIGMOD Rec. 31 (3) (2002) 9–18.
- [35] B. L. Gorman, M. Shankar, C. M. Smith, Advancing Sensor Web Interoperability, Sensors Magazine 22 (4) (2005) 14–18.
URL <http://www.sensorsmag.com/sensors/article/articleDetail.jsp?id=185897>
- [36] G. Percivall, C. Reed, OGC Sensor Web Enablement Standards, Sensors and Transducers 9 (9) (2006) 698–706.
- [37] K. B. Lee, M. E. Reichardt, Open Standards for Homeland Security Sensor Networks, IEEE Instrumentation & Measurement Magazine 8 (5) (2005) 14–21.
- [38] A Smart Transducer Interface for Sensors and Actuators, IEEE Draft Std. (2007).
- [39] UPnP Device Architecture (2006).
- [40] Computational Sciences and Engineering Division, SensorNet: Concept Definition Document, Tech. report, Oak Ridge National Laboratory (2004).

- [41] G. J. Pottie, W. J. Kaiser, Wireless Integrated Network Sensors, Communications of the ACM 43 (5) (2000) 51–58.
- [42] J. P. Vasseur, Routing Over Low Power and Lossy Networks (roll), IETF Working Group (Dec. 17 2007).
URL <http://www.ietf.org/html.charters/roll-charter.html>
- [43] Q. Huaifeng, Z. Xingshe, Context Aware SensorNet, in: MPAC '05: Proceedings of the 3rd International Workshop on Middleware for Pervasive and Ad-hoc Computing, ACM Press, New York, NY, USA, 2005, pp. 1–7.
- [44] A. Rezgui, M. Eltoweissy, Service-Oriented Sensor-Actuator Networks, IEEE Communications Magazine 45 (12) (2007) 92–100.
- [45] M. Marin-Perianu, N. Meratnia, P. Havinga, L. M. S. D. Souza, J. Mller, P. Spiess, S. Haller, T. Riedel, C. Decker, G. Stromberg, Decentralized Enterprise Systems: a Multiplatform Wireless Sensor Network Approach, IEEE Wireless Communications [see also IEEE Personal Communications] 14 (6) (2007) 57–66.
- [46] Y. Qian, K. Lu, D. Tipper, A Design for Secure and Survivable Wireless Sensor Networks, IEEE Wireless Communications [see also IEEE Personal Communications] 14 (5) (2007) 30–37.
- [47] I. Saleh, A. Agbaria, M. Eltoweissy, In-network Fault Tolerance in Networked Sensor Systems, in: DIWANS '06: Proceedings of the 2006 Workshop on Dependability Issues in Wireless Ad Hoc Networks and Sensor Networks, ACM, New York, NY, USA, 2006, pp. 47–54.
- [48] I. Chisalita, N. Shahmehri, A Peer-to-peer Approach to Vehicular Communication for the Support of Traffic Safety Applications, in: Proceedings of the IEEE 5th International Conference on Intelligent Transportation Systems, 2002, pp. 336–341.

- [49] T. Nadeem, S. Dashtinezhad, C. Liao, L. Iftode, TrafficView: Traffic Data Dissemination Using Car-to-Car Communication, SIGMOBILE Mobile Computing and Communications Review 8 (3) (2004) 6–19.
- [50] M. C. Edwards, J. Donefson, W. M. Zavis, A. Prabhakaran, D. C. Brabb, A. S. Jackson, Improving Freight Rail Safety with on-board Monitoring and Control Systems, in: Proceedings of the 2005 ASME/IEEE Joint Rail Conference, 2005, pp. 117–122.
- [51] S. Muralidharan, V. Frost, G. J. Minden, SensorNet Architecture with Multiple Owners, Tech. Report ITTC-FY2008-TR-41420-02, University of Kansas, Lawrence, Kansas (July 2007).
- [52] G. J. Minden, J. B. Evans, A. Agah, J. W. James, L. Searl, Architecture and Prototype of an Ambient Computational Environment: Final Report, Tech. Report ITTC-FY2004-TR-23150-09, University of Kansas, Lawrence, Kansas (July 2003).
- [53] M. Botts, G. Percival, C. Reed, J. Davidson, OGC Sensor Web Enablement: Overview and High Level Architecture, OGC 06-050r2, Architecture (2006).
URL <http://www.opengeospatial.org/pt/06-046r2>
- [54] M. Botts, OpenGIS Sensor Model Language (SensorML) Implementation Specification, Specification (2005).
URL http://portal.opengeospatial.org/files/?artifact_id=13879
- [55] J. Mauro, Security Model in the Ambient Computational Environment, Master’s thesis, University of Kansas (2002).



Technical Report

A Survey on Methods for Broadband Internet Access on Trains

Daniel T. Fokum and Victor S. Frost

ITTC-FY2009-TR-41420-09

August 2008

Project Sponsor:
Oak Ridge National Laboratory

A Survey on Methods for Broadband Internet Access on Trains

Daniel T. Fokum, Student Member, IEEE
Victor S. Frost, Fellow, IEEE

Abstract

Interest in broadband Internet access on trains has been increasing. Here a survey of approaches for providing broadband Internet access to trains is presented. In this paper we examine some of the factors that hinder the use of broadband Internet on trains, and then examine some of the opportunities for broadband deployment on trains. This survey examines some of the basic concepts for providing broadband Internet access and then reviews associated network architectures. The review of network architectures shows that we can subdivide networks for providing broadband Internet access on trains into the train-based network, the access network — for connecting the train to the service provider(s), — and the aggregation network for collecting user packets generated in the access network for transmission to the Internet. Furthermore, our review shows that the current trend is to provide Internet access to passengers on trains using IEEE 802.11x; however, a clear method for how to connect trains to the global Internet has yet to emerge. A summary of implementation efforts in Europe and North America serves to highlight some of the schemes that have been used thus far to connect trains to the Internet. This paper concludes by discussing some of the models developed, from a technical perspective, to test the viability of deploying Internet access to trains.

1 Introduction

With the explosion in growth of the Internet in the last 20 years, people have a much higher expectation of being able to get on the Internet independent of location. Until recently trains and airplanes have been two locations where passengers have not necessarily been able to achieve high-speed Internet connections. In the particular case of trains, providing Internet access to passengers on board trains makes good business sense; Internet access for passengers can provide a revenue stream for the train company while attracting more travelers. For example, a 2004 study in the United Kingdom found that 72% of business travelers were more likely to use trains than cars or airplanes if Wi-Fi access was available on trains. This study also found that 78% of business travelers would use Wi-Fi access if it was made available on trains [1]. In the case of freight trains Internet access can allow for real-time or near-real-time tracking of freight-related events on board the train, and potentially resulting in a decrease in insurance charges to the freight carrier. In addition to these benefits, on-board Internet access on trains can also enhance the safety of the train, by allowing an operations center to monitor, in real-time, train-related data, as in [2].

Internet access on board trains is already available today in parts of Europe. For example, beginning in July 2004 a British train operator, GNER¹, began offering Internet access on some of its trains [3]. In 2005 another British company, Nomad Digital, claimed to have addressed the problem of providing high-speed Internet access to passengers on Southern Trains' London to Brighton route using WiMax [4]. In what follows we provide an overview of communications on board trains, beginning with some of the earliest papers on providing high-speed Internet access to users on the move.

The main contribution of this paper is to provide a survey of work done on providing Internet access to trains. The conditions of a rail environment that make communications from trains difficult are highlighted. Another contribution of this paper is to summarize projects that seek to provide Internet connectivity on board trains. For reasons that shall become apparent later, we make a distinction between work done in Europe, and work done in North America due to the different characteristics of rail transportation on those continents. The rest of this paper is laid out as follows: Section 2 lists the issues hindering high-speed communications from trains. In Section 3 we present a reference architecture for the deployment of broadband Internet access to trains. Section 3 also presents the initial concepts for broadband Internet access on trains. In Section 4 we present the efforts made, or those efforts underway to carry out high-speed communications from trains. Section 4 is further subdivided into examining implementation efforts underway in Europe and in North America. Section 4 concludes by presenting business models developed to determine the viability of providing broadband Internet access on trains. In Section 5 we provide concluding remarks.

2 Difficulties and Opportunities

2.1 Difficulties

Communications on board trains are complicated by several factors. Lannoo *et al.* [5] state that railcars have Faraday cage like characteristics which can lead to high penetration losses for signals. Beeby [6] adds that some other complicating factors include:

- A high vibration environment that may require mechanical isolation of communication devices
- A thermally challenging environment, since heat may be a significant issue in certain parts of the train
- A harsh electrical environment due to:
 - The proximity of high voltages, as in electrical trains
 - High magnetic fields, as in magnetic levitation (Maglev) trains
 - Trains are not designed to provide a 'clean' electrical supply for computers
- The need to have equipment with minimal maintenance schedules – this may result in equipment with near military-grade specifications
- The presence of track-side features, such as railway signaling equipment

Some other factors hindering communications on trains include:

¹GNER subsequently lost its license to operate the East Coast Mainline, where the Wi-Fi-enabled trains were deployed. National Express replaced GNER on the East Coast Mainline, and they offer free wireless (Wi-Fi) Internet access on all trains on the East Coast line.

- Railway companies constantly add or remove rail cars from trains. As a result it is necessary for the communications network to discover these changes automatically [7].
- Poor coupler contacts on rail vehicles, which may introduce communications failures [7].
- Tunnels may limit visibility to wireless communication infrastructure.
- Frequent handoffs in the cellular network. To see why this might be a problem, consider a train travelling at 60 m/s (216 km/h) through an environment with cell sizes of the order of 3 km, then we would have handoffs every 50 s. Assuming that it takes about 1 s to complete a handoff the handover time is 2% of the dwell time in the cell, which is high.

In spite of these difficulties, there are several opportunities to provide Internet access on trains using a variety of technologies, including Wi-Fi, WiMax, satellite technologies, and radio-over-fiber. In Section 2.2 we discuss some of these opportunities.

2.2 Opportunities

The growth in wireless communication technologies over the last two decades opens up several opportunities for supporting communication on board trains. For example, customers in a stationary train can have Internet access through the existing cellular infrastructure without many modifications, except for an antenna on the outside of the train. Issues only arise when the train begins to move, particularly at high speeds and requires several handoffs in a short period of time. Beeby [8] argues that communications capabilities on mobile terminals is constantly improving, with some phones now having multiband and Wi-Fi capabilities. Currently it is standard to have Wi-Fi integrated on laptops, and eventually WiMax might also be commonly available. These factors, especially the latter, have the potential to drive Internet usage higher, particularly because as connectivity becomes more prevalent usage increases [8]. Beeby goes on to argue that there are significant opportunities available for Internet access on trains if [8]:

- access to the technology can be made simple,
- ubiquitous (as in not requiring any special software or terminal), and
- useable, i.e., acceptable throughput and delay with few service interruptions.

In this respect, Fourth Generation (4G) communications technologies, such as WiMax, offer the best potential for offering Internet access on trains. In fact, WiMax is already being used in the UK to provide Internet access on trains [4, 9-11]. We expect further growth in broadband Internet access availability on trains as more train operators are convinced of the business viability of negotiating for Internet access along their tracks using WiMax or some other 4G technology.

3 Reference Architecture and Initial Concepts

In this section we present a reference architecture to guide our discussion of broadband Internet access on trains. Next we examine some initial ideas related to broadband Internet deployment on trains.

3.1 Reference Architecture

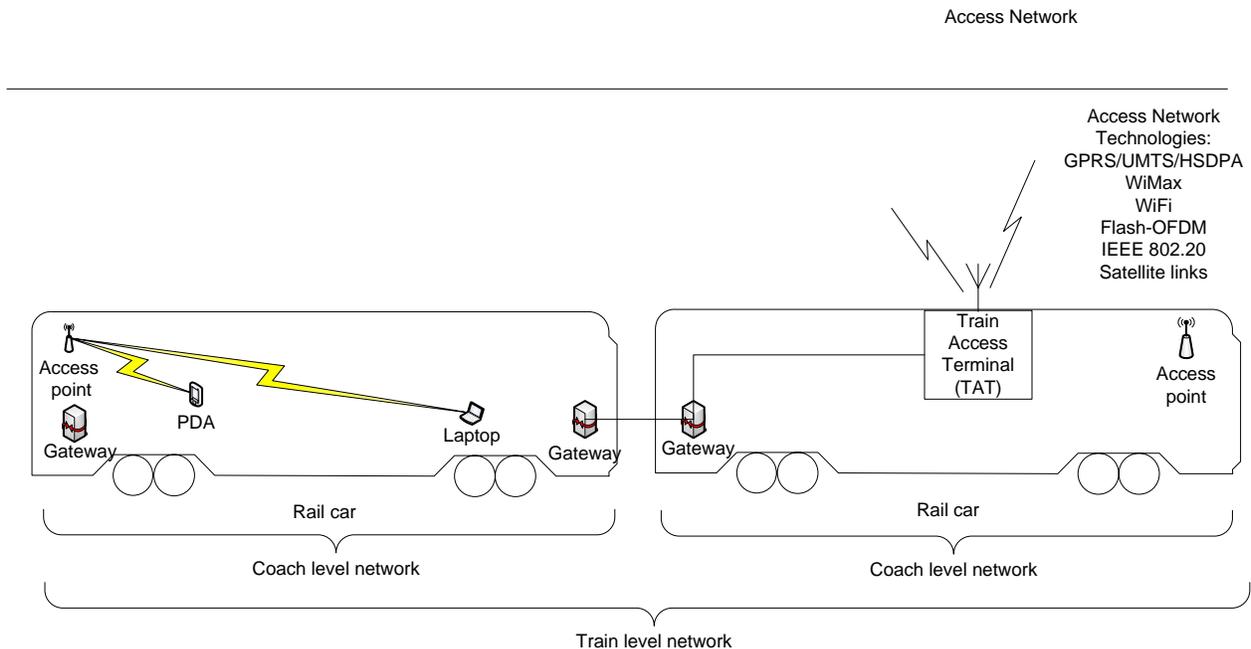


Figure 1: Architecture for Internet Connectivity between Rail Cars

Fig. 1 shows a logical architecture for the computer networks on a train, i.e., the networks aboard the trains used to provide access to passengers. This architecture, which incorporates aspects of the Train Management System [12], uses gateways in each train car to build a train level network. Broadband Internet access on the train is provided through the Train Access Terminal (TAT). This terminal, which can support one or many technology types, connects to the access network using an antenna mounted on the outside of one train car. The incoming signal from the train access terminal is then fed to gateways and wireless access points in all the rail cars in the train. Within each rail car IEEE 802.11 is commonly proposed to provide connectivity to passengers; however, passengers may also connect to a wired network in the railcar, if one is available. The benefit of using an architecture such as the one described above include the following:

1. The cellular network system is not put under strain attempting to make handovers for several fast moving users simultaneously [13].
2. The train access terminal can combine different access technologies for network redundancy. In addition, the train access terminal can also implement some “intelligence” to select the best means of communication between the train and the access network, as in [13].

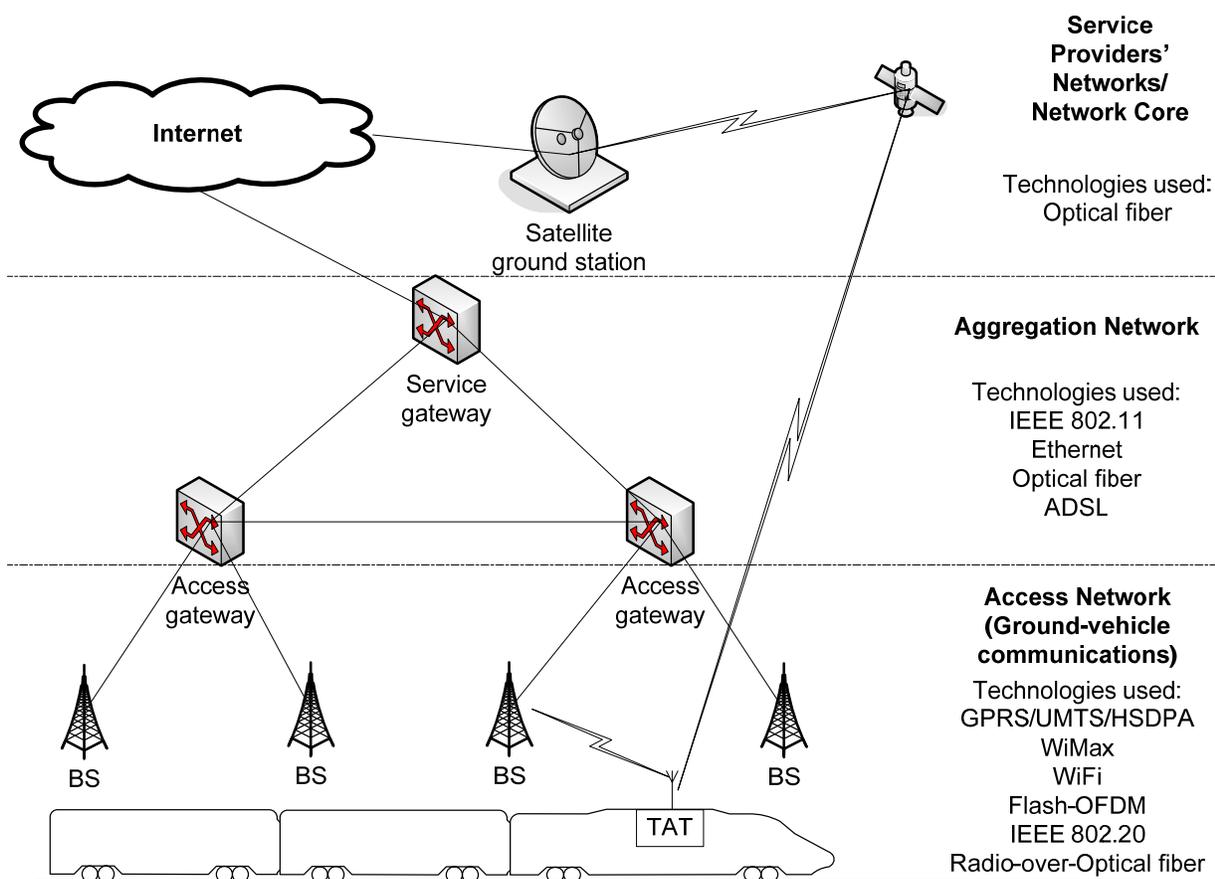


Figure 2: Reference Architecture for Internet Access on Trains

The entire train is connected to the Internet using the reference architecture shown in Fig. 2. The reference architecture for Internet access on trains is layered and consists of the access network, aggregation network, and the service providers' networks. The access network is close to the train tracks, and it provides the last hop communications for the train access terminal. The aggregation network lies between the access network and the service providers' networks, and it forwards data from the access network to the global Internet. The access gateway in the architecture combines the data from a group of users into a tunnel and forwards that data to the service gateway. The service gateway serves as an interface between the aggregation network and service providers' networks. Van Quickenborne *et al.* [14] argue that aggregated tunnels per train are ideal for this architecture since they are more manageable and efficient than a per user connection scheme. From the reference architecture diagram, we can also see that there are different technology options, including satellite technologies, for the access and aggregation networks. This observation is in agreement with Conti [10], who states that currently there is general agreement on how to provide Internet access to passengers aboard trains. A disagreement arises on the best method to connect moving trains to the Internet backbone, i.e., how to connect the antenna on the train access terminal to the access network. However, we expect that the

widespread deployment of 4G technologies may lead to some consensus on the best way to connect trains to the Internet.

It should be noted that Fig. 2 combines features of several proposed architectures, including the FAMOUS² architecture [15] that we will see later in this paper. Some other features of this architecture include:

- The access network is a wireless network with base stations along the train tracks. The access network can use either GPRS/UMTS/HSDPA [9, 10], WiMax [4, 9-11], WiFi [16], Flash-OFDM [17], satellite links [10], or IEEE 802.20 [18].
- The aggregation network can use the following technologies for forwarding data: IEEE 802.11 [19, 20], Ethernet [14, 21-24] and Radio-over-fiber [5, 25].
- Virtual Local Area Networks (VLANs) are preferred in Ethernet-based aggregation networks to carry aggregated traffic flows from the access gateway to the service gateway [15]. In other words, VLANs are used in the aggregation network to group the different base stations in an access network that satisfy a given train's traffic demands.
- Satellite links³ can be used to provide Internet access to trains; however, they do not fit this architecture neatly, since the satellite ground station cannot be easily classified as either a service gateway or an access gateway. Consequently the satellite links in Fig. 2 are shown straddling the different networks.

3.2 Initial Concepts

Efforts to deploy broadband Internet access on board trains are constantly evolving. In this subsection we provide an overview of some of the initial concepts that have guided the deployment of broadband Internet access on trains.

Due to their mobility, Internet access can be provided on trains only by use of a wireless link. Correia and Prasad [26] present some of the technical challenges involved in providing wireless broadband services. The reader is referred to [26] for a more complete treatment of the important attributes of a wireless broadband system. The problem of providing broadband communications to fast moving users was addressed in [27] and [28]. In 2001 Gavrilovich [27] argued that a large number of small cells operating at high frequencies was the most economical and practical infrastructure for providing wireless broadband access to many users. In Gavrilovich's model these small cells were provided by moving base stations that travel along a track beside the roadway. The moving base stations were then linked to fixed base stations using wireless links. The fixed base stations were uniformly distributed along the roadway and were also interspersed with the mobile base stations. This combination of mobile and stationary base stations allowed the realization of broadband wireless communications while also yielding fewer handoffs due to the mobile base stations [27]. However, a moving base station may not be practical.

An architecture for providing communications and entertainment aboard a high-speed public transport system is proposed in [28]. This architecture is composed of the following components:

- A mobile subsystem that consists of mobile subnetwork, access to an infrastructure network, and a mobility management component. This mobile subsystem is analogous to the access network in this paper; however it does not include any of the wireless communication technologies incorporated at the access network in our architecture.

²The FAMOUS architecture was developed to provide Internet access to FAst MOving USers

³Lannoo *et al.* [5] state that satellite communications are not ideal for high-speed access to trains since satellite links have limited bandwidth and long round trip times (RTT).

- A wireless transport subsystem that handles radio transmission between the mobile subsystem and the infrastructure network. This subsystem is analogous to the wireless communication technologies found in the access network of our reference architecture.
- A land subsystem consisting of an infrastructure network and a network management component. This would be analogous to the aggregation network and the service providers' core networks.

In addition Lin and Chang [28] argue that the link between the passenger device and the base station can be provided by IEEE 802.11, Bluetooth, or one of the Third Generation (3G) wireless standards. As we have seen already, WiMax — which is one of the 4G wireless standards — has been chosen in [4, 9] to provide the link between the train and a terrestrial network, while Wi-Fi has been chosen to provide the link between the passenger terminal and the train network. Finally, [28] also notes that for a train moving along a track, the cell planning problem reduces to a one-dimensional problem, which should greatly facilitate frequency planning.

Next we examine work done on the FAMOUS architecture; an architecture designed to support broadband Internet access for FAsT MOving USers. All of this work ([5, 12-15, 19-25, 29-34]) was conducted by researchers in Belgium.

3.2.1 FAMOUS Architecture and Its Extensions

In 2003 it was observed that popular Internet applications may not be available at high-speeds due to lack of bandwidth, poor quality of service, and frequent hand-offs [29]. These problems could be partially addressed by: increasing network bandwidth using smart antenna systems and MIMO technologies, as well as improved handoff protocols that prevent connection loss when moving from one base station to another. Van Leeuwen *et al.* [29] state that the technologies discussed above are not sufficient to support broadband communications at high speeds; new modulation schemes and context-aware applications are also needed to achieve high data rates in fast moving vehicles.

Other ideas for supporting broadband communications from trains divide such a network into an access network and an aggregation network [15] and [21]. Each of these networks performs the functions that were discussed in Section 3.1. Furthermore, each network is located as previously described. In the FAMOUS architecture users do not connect directly to the base stations in the access network; instead the entire train has a single connection to the access network. This connection is then shared amongst all the users on the train. The FAMOUS architecture assumes that seamless connectivity is not guaranteed for users in fast moving vehicles; instead they will hop from one access gateway (AGW) to the next [24]. Within the aggregation network VLANs are used to group the different base stations in an access network that satisfy a given train's traffic demands [15]. Another component of this network architecture is the service gateway, where connections between service providers and the aggregation network are made. The FAMOUS architecture is summarized in Fig. 3.

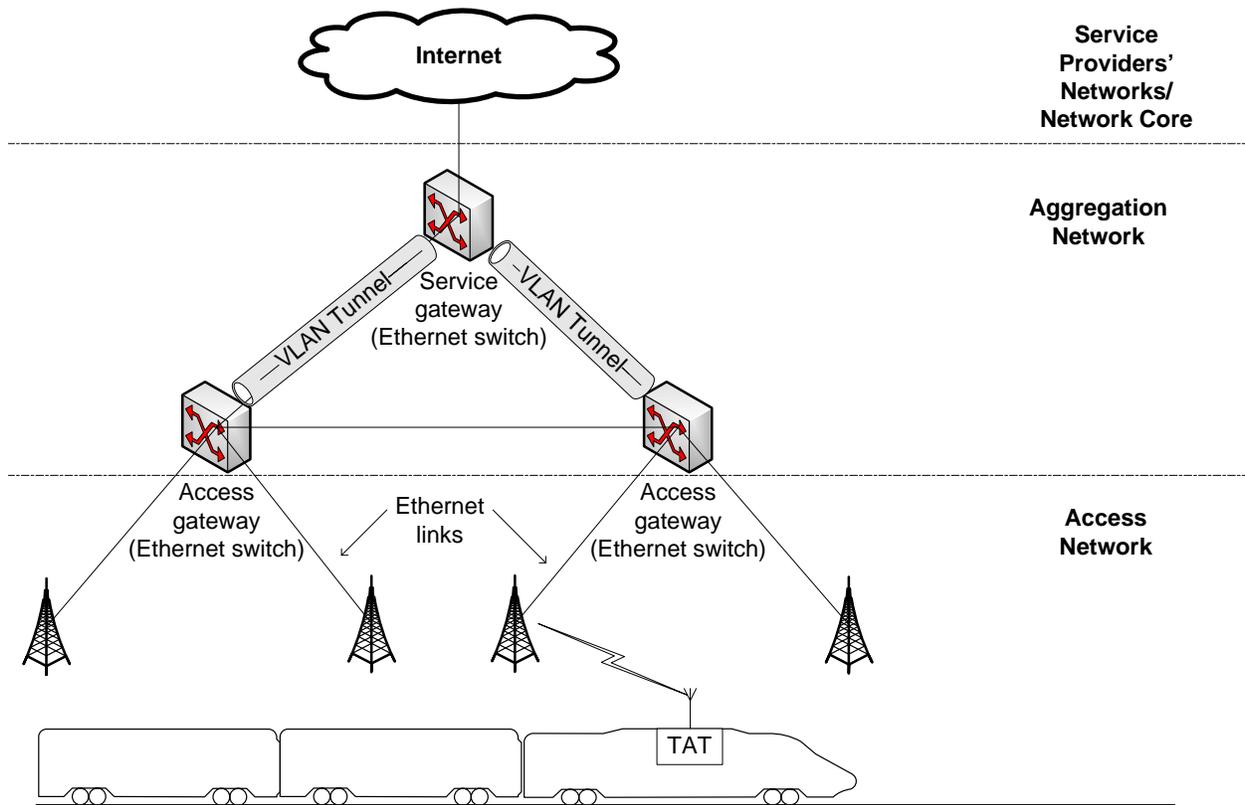


Figure 3: FAMOUS Architecture for providing broadband Internet access to FAst MOving Users based on [15, 23]

In [15, 21] and subsequent papers [14, 22-24, 31, 32], Ethernet is chosen for the aggregation network since it is simple, cost-effective and bandwidth flexible. In spite of Ethernet's advantages, it requires some modifications to support fast moving users. One of Ethernet's issues is the rapid depletion of VLAN databases in Ethernet switches. In fact, Ethernet already has an extension, called GVRP⁴ [21], that can register VLANs automatically in a consistent and reliable manner; however, standard GVRP distributes VLAN IDs of all tunnels to all the switches in the network, thereby flooding the VLAN databases. This issue is resolved by developing a "scoped refresh" of GVRP, such that Ethernet switches determine whether or not they are part of a given tunnel. If they are, then the switch will issue de-registration messages on all its interfaces that do not have the VLAN registered, otherwise the switch would attempt to register the VLAN. GVRP needs a mechanism to configure Ethernet switch hardware to meet the QoS-parameters associated with each tunnel. As a result G2RP⁵ has been developed to support fast moving users by allowing for the separate distribution of traffic reservation parameters and VLANs. When combined GVRP and G2RP allow switched Ethernet to be used as a transport technology for an aggregation network.

⁴GARP VLAN Registration Protocol, where GARP is Generic Attribute Registration Protocol

⁵GARP Reservation Parameters Registration Protocol

The same architecture used in developing G2RP is also found in [15]; however, [15] focuses on handoff strategies in a network with fast moving users found in cars or trains. In 2005 De Greve *et al.* [15] stated that high link speeds for end users could only be achieved in cellular networks by reducing the cell size to efficiently reuse spectrum. However, small cells also mean more handoffs between cells. Furthermore, Mobile IP is not a good protocol for delivering high link speeds to fast moving users since Mobile IP does not work well with frequent handoffs due to handoff latency, handoff packet loss and control message load. As a result, [15] stated that higher link speeds could be given to fast moving users on a train by using small cells operating in the millimeter wave band. In addition these authors suggest using radio-over-fiber with moveable cells to reduce handoff times, an idea that is an extension of Gavrilovich's moving base stations model [27]. We will revisit this concept later in this subsection.

Handoffs may result in packet loss, consequently handoffs, i.e., when the train access terminal hops from one base station in the access network to the next, must be tackled satisfactorily to provide broadband communications to fast moving users. De Greve *et al.* [23] presents the Motion-aware Capacity and Flow Assignment (MCFA) algorithm to optimize the use of network resources, determine paths for dynamic tunnels in an aggregation network, and minimize the impact of packet loss and packet reordering when designing an aggregation network to support fast moving users. The schemes presented include:

- An ideal routing algorithm for minimal network cost, which does not take any additional constraints into account when solving the MCFA problem.
- A limited Hop Count Variations routing scheme, which guarantees maximum delay by limiting the variation in hop counts between two different paths.
- A shared routing algorithm, which requires the paths assigned to a given connection between the node and the aggregation network to share some nodes in common.
- An incremental routing method, which is an even stricter form of shared routing in which the different paths share even more nodes in common.

Of the schemes presented in [23], incremental routing exhibits excellent packet loss features but poor scalability while Limited Hop Count Variations (LHCV) routing yields a network that has a slightly higher network cost than if ideal routing was used. However, LHCV routing shows better congestion performance. De Greve *et al.* [23] also presents a heuristic, called Subpath Assignment (SpA), for mapping aggregation network routes onto a minimal set of spanning tree instances. When this heuristic is compared with other path aggregation schemes it is seen that SpA can perform the path mapping in the shortest amount of time.

Recall that in the aggregation network in the FAMOUS architecture is built on switched Ethernet. Furthermore, dynamic tunnels are used in the aggregation network to support the traffic demand from a given set of trains. For switched Ethernet to be used in a carrier-grade network, Ethernet must provide a mechanism for fast recovery from link failures in the aggregation network. De Greve *et al.* [24] present an extension to Ethernet's Rapid Spanning Tree Protocol (RSTP) that uses a fast detection mechanism for link and node failures. This mechanism, which is resilient to node or link failures, bypasses the RSTP failure detection process and monitors links by examining incoming and outgoing packets at a given switch. De Greve *et al.* [24] show that if reliability constraints are added to the MCFA optimization problem, then it is possible to have good recovery times in the aggregation network, even when there are dynamic VLANs present.

De Greve *et al.* [22] argue that aggregation networks are not optimally designed for broadband services from fast moving vehicles, e.g., trains; therefore, it develops an integer linear

program (ILP) to calculate the exact dimensioning and tunnel paths needed to satisfy traffic demands from a train to the global Internet. For large network cases, the ILP can take several days or weeks to solve; therefore, De Greve *et al.* [22] develop and apply a heuristic — which achieves low congestion and optimizes the use of network resources — to solve the problem, i.e., meeting the traffic demands of fast moving users in the FAMOUS architecture. In the ILP model each train is assumed to generate a certain amount of traffic, where these traffic demands can be defined as either:

- Exact, which would require optimization of network resources with knowledge of the exact access gateway (AGW) where two trains cross each other, and the exact instant when the crossing occurs.
- Static, which results from exact demands by neglecting all time-related aspects of the demand. This is required if a network lacks a dynamic reservation mechanism; however, it results in over-dimensioning of resources.
- Train delay insensitive (TDI), which results from neglecting the exact time-position between multiple trains, i.e., we neglect the information of exact point when and where the trains cross each other. This implies the network is dimensioned to allow for trains to cross at any AGW along their respective paths.

It is shown in [22] that using TDI demand results in a more complex optimization problem; however, if traffic demands are defined as train delay insensitive, the QoS guarantees of passengers can be fulfilled always. In addition [22] concludes that for optimal network design the links that need to be considered for connecting the service gateway to the access gateway are those closest to the rail line end terminuses⁶

Van Quickenborne *et al.* combines the findings from [21] and [22] in [31]. Reference [31] deals with designing an aggregation network that combines data from several users as they move from one access network to the next. The access network traffic is aggregated into tunnels in the aggregation network, and these tunnels have to move with the users from one access network to the next. In designing the aggregation network Van Quickenborne *et al.* [31] relies on an objective function that minimizes the number of hops between the train and the service gateways. The objective function's constraints include link capacity constraints and ensuring that only one path is needed from source to destination. Using this optimization model it can be shown that if each train requires two dynamic tunnels — one for basic demand and the other tunnel for transient spikes in traffic demand — then the solution to the optimization problem can be obtained quickly. On the other hand, this problem takes longer to solve if we seek to minimize the cost of the network interface card and routing costs subject to the same constraints. Another result from this paper shows that dynamic tunnel configuration and activation reduces network cost, since the basic traffic demand is routed over a shorter path, while the transient spikes in traffic demand are routed over longer paths [31].

The FAMOUS architecture has also been used in [14] to show that a hierarchical wired Ethernet aggregation network in combination with Ethernet-based⁷ wireless access networks

⁶In this problem assume that the different towns/stations in the rail network represent the vertices of a graph, while the rail lines represent the edges of the graph. Then, only links between the service gateway and the access gateways closest to the vertices need to be considered when using the heuristic approach. For more details please consult [22].

⁷Reference [14] presents an example of an Ethernet-based wireless access network that has a single WiMax station per access network. Each base station is linked to the aggregation network via an Ethernet link.

between trackside antennas and the train access terminal may be used for providing broadband Internet access to fast moving users. Reference [14] assumes the use of dynamic tunnels, as proposed in [31]. Here the dynamic tunnel management takes one of three forms:

- Management-based approach, that uses location information, e.g., from GPS to set up tunnels to a train. When the train arrives at an access gateway, the train's location information is sent to a management platform that sets up the train's tunnels. When the train moves to another access gateway the previous tunnel is torn down.
- Signaling-based approach, in which a train announces its presence at a given access gateway, resulting in tunnel setup for the train. After a timer expires the tunnels are torn down.
- Hybrid approach, which incorporates portions of the schemes described above, i.e., a signaling-based approach in the tunnels nearer the train, and a management-based approach in the higher parts of the network.

Simulation results from [14] show that the signaling-based approach is hard to use in aggregation networks, since tunnel-setup times increase with tunnel length — number of hops in the aggregation network. As a result, the hybrid approach is recommended. This approach has the added benefit of reducing packet loss while providing accurate tunnel-setup triggers.

The FAMOUS architecture is also extended in [19] and [20] to support the case where several leaf nodes (trains) require connectivity with a limited set of service gateways through a wireless mesh network, i.e., the aggregation network is built using wireless mesh networks. De Greve *et al.* [19] say this is possible because wireless mesh networks are cheaper to deploy than their wired equivalents. In [19, 20] the access gateways are replaced by wireless gateways. In addition the underlying aggregation network technology is replaced with IEEE 802.11e instead of switched Ethernet [19]; in the future we expect that such a wireless aggregation network can also be provided by emerging Ethernet-based gigabit radios. Wireless networks can sometimes be subject to reduced throughput due to interference from neighboring stations. Therefore, De Greve *et al.* [20] suggest wireless throughput may be improved in mesh networks by intelligent distribution of neighbour mesh nodes and minimising link interference levels by assigning different channels to the different interfaces of the wireless gateways. These objectives can be achieved by using a distributed channel assignment module that tries to minimise interference levels on links by assigning different channels to various interfaces on the wireless gateways. Fast moving users can then be supported by using a wireless mesh node placement algorithm that minimizes the hop count of the service gateway-wireless gateway paths [20].

In [32] the FAMOUS architecture is used to provide high-bandwidth and low latency traffic to fast moving users. In this case the MCFA optimization problem from [24] is used to determine optimal aggregation gateway location, the number and speeds of interface cards, and traffic tunnel set-up. The routes computed by MCFA are then mapped onto VLANs and spanning tree instances for routing in the FAMOUS architecture. Results from a testbed show that low latency high bandwidth links can be provided to fast moving users, and that rapid recovery with spanning trees is feasible without a centralized system [32].

In 2005 and 2007 Lannoo *et al.* ([5] and [25]) proposed extensions to Gavrilovich's [27] moving base stations model. Lannoo *et al.* [5] argue, just as in [15], that frequent handovers greatly reduce the bandwidth available to fast moving users. Consequently they propose using radio-over-fiber, as suggested in [15], to feed base stations along the rail track. Unlike in Gavrilovich's model there are no moving moving base stations; instead there is a fiber-fed distributed antenna network. These distributed antennas are located along the railroad tracks, and

they are called remote antenna units (RAU) (these correspond to the base stations in Fig. 2). The remote antenna units are supervised by one control station via an optical ring network. For communications from the access network to the train, data is modulated at the control station and sent optically to each remote antenna unit using wavelength division multiplexing, i.e., each RAU has a unique wavelength for communications. The remote antenna unit will convert the optical signal to radio waves and transmit to the train. For communications from the train to the access network the data will typically be captured by the remote antenna unit closest to the train. In order to reduce handover times for the train access terminal, Lannoo *et al.* propose using “moving cells”, i.e., a cell pattern that is constantly reconfigured at the same speed as the train so that the train access terminal communicates on the same frequency during a trip. For a more complete treatment of Lannoo’s moving cell concept, please consult [5]. Fig. 4 presents a reference architecture for the radio-over-fiber deployment.

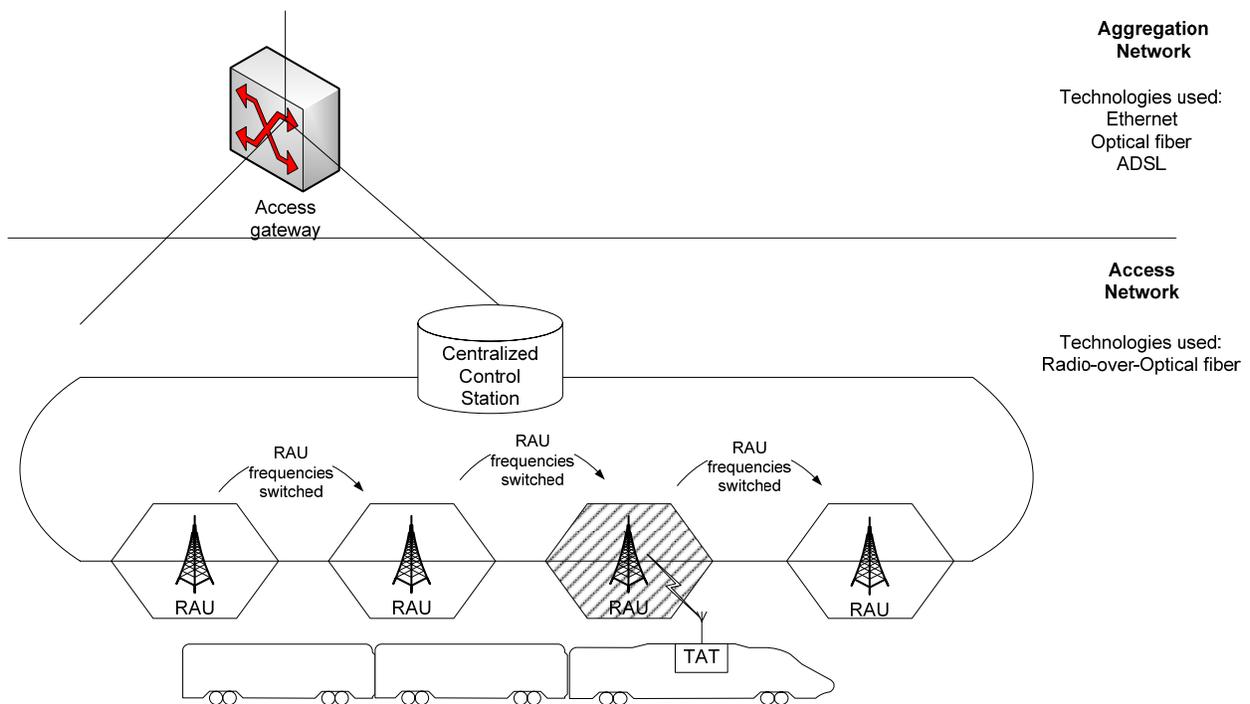


Figure 4: Reference Architecture for Internet Access on Trains using Radio-over-Fiber, based on [5]

3.2.2 Other Architectures, Handoff and Addressing Issues

An architecture similar to the FAMOUS architecture is found in [35]. This architecture divides train communications into backhaul connections, Ground-to-vehicle communications (GVC) and on-board vehicle communications (OVC). The GVC is analogous to the access network in our reference architecture in Fig. 2, while the OVC network consists of customer devices as well as other networking devices, such as a train server, placed in the train. The OVC network is similar to the train-based network shown in Fig. 1. On board each train the OVC and GVC are connected through a connection manager (CM), which is analogous to our train access terminal in Fig. 1 [35].

In 2003 Bianchi *et al.* [36] thought that it may be expensive to wire a train for network access. In addition [36] stated that rewiring may be needed every time the train is reconfigured. Therefore, they proposed using IEEE 802.11 to construct a wireless network between the train cars. In their basic architecture, the train is connected to the Internet through a “train server” using satellite links. The train server here is analogous to the train access terminal in Fig. 1. Aboard the train IEEE 802.11 is used to perform the following tasks:

1. To link all the railcars on the train into a computer network,
2. To provide Internet access to passengers, and
3. To connect the train to the Internet when the satellite links become too expensive. For example, if a given train station has IEEE 802.11 access points, the train can be connected to the Internet through those access points instead of through the train server.

Bianchi *et al.* proposed two topologies, based on IEEE 802.11 for constructing the computer network aboard the train. In their first topology the railcars are linked into a network using IEEE 802.11 access points whose antennas are on the outside of each railcar, i.e., in this case the gateways shown in Fig. 1 are IEEE 802.11 access points. In order to minimize interference between adjacent access points, Bianchi *et al.* state that directional antennas should be used in this deployment. Furthermore, channels should be chosen on each access point, such that neighboring access points do not interfere with each other. Additional gains in performance may be achieved by using IEEE 802.11a for the wireless network between railcars, and IEEE 802.11b within the rail car. These technology choices imply that the computer network on the outside of the train would not interfere with that inside the railcars. An alternative topology for the network aboard the train arranges the access points in each railcar such that each access point serves as a client station for the access point in the previous car, while also serving as an access point for all the stations within its car. In other words, given train cars 1 and 2; the access point in car 2 serves as a client (station) of the access point in car 1 while also serving as the host (access point) for all stations within car 2. Since an access point may not transmit and receive simultaneously, this topology requires that each access point possess two interface cards – one for transmitting and the other for receiving. Bianchi *et al.* conclude by noting that their proposed topologies need to be tested in a real-world deployment to assess the impact of interference [36].

References [12, 13, 33] come from the same group that developed the FAMOUS architecture, but these papers do not use that architecture directly. Jooris *et al.* [33] studies seamless handover, roaming, Quality of Service (QoS), and connections between heterogeneous wireless networks, such as the on-board network and the track-side network. On each train the Mobile Access Router (MAR) — for connecting the train to the outside Internet — will have one interface for each type of technology, and it will constantly choose the best link from the train to the outside world. It should be observed that the mobile access router is analogous to the train access terminal (TAT) in Fig. 1. Aboard a train handoffs can occur when a mobile device is either unplugged from the train’s wired network or when a mobile user moves from one Wi-Fi hotspot on the train to another. In each case the user’s session must be protected. Jooris *et al.* [33] proposes carrying out this protection by creating a convergence layer that hides the Ethernet and WLAN interfaces, and instead creates a single virtual interface and assigning a single IP (CL-IP⁸) and a single MAC (CL-MAC⁹) address to it. Outgoing packets will be encapsulated with the CL-IP and CL-MAC, while devices connected to the train LAN will only see one device and one

⁸Convergence Layer IP

⁹Convergence Layer MAC

MAC address. In Jooris *et al.*'s implementation [33] every wireless user device is associated with a unique software object, which they call the access point. This software object is installed on the nearest base station (BS) on the train, but it is moved from one WLAN base station to the next as the user moves. In this architecture each base station is configured with two interfaces, but the BS operates on a fixed frequency. The first interface runs an access point for all WLAN stations — for example, wireless user devices — within range of the BS, whereas the second interface listens to neighboring base stations' frequencies and measures the signal strengths of the broadcast messages. If the second interface detects a stronger signal from a station than the signal measured by the station's current base station, then the station's access point is changed to that of the measuring interface. The station is also informed that its access point has changed frequency. This handover mechanism has been simulated successfully, and it should allow passengers to be mobile while using the networks on board trains.

Pareit *et al.* [13] assumes that one would need to combine different technologies to provide broadband Internet access on trains. As a result, they tackle the issue of handoffs as the train moves from the coverage area of one access technology to another in [13]. To prevent the cell system from having to make several simultaneous handoffs, it is proposed that train passengers connect to the Internet via on-board Wi-Fi access points that are connected to the local train network. The architecture proposed in [13] places a Policy Decision Function (PDF) on the gateway, i.e., the train access terminal, between the train's network and the outside world. The PDF decides which interface should be used to provide the connection between the train and the access network. This decision is based on link quality, train location and speed, and possibly cost or load balancing. The Mobility Management modules are the other key part of the architecture. They reside partly on the train and partly on the Central Management System. These modules take input from the PDF to make handovers as smooth as possible. Pareit *et al.* [13] evaluates the feasibility of using either Mobile IP or MMP-SCTP¹⁰ for a mobility management handoff protocol. Recall that Mobile IP allows nodes to change their point of attachment to the Internet without changing their IP address [13], while Stream Control Transport Protocol (SCTP) is a reliable transport protocol that resides above an unreliable connectionless packet service [13]. SCTP allows for the detection and retransmission of packets that might be lost during a handover. In addition SCTP endpoints allow for multihoming. In [13] it is shown that MMP-SCTP displays better performance than Mobile IP after a slow start for TCP performance without a handoff. Pareit *et al.* emulate the case where a train passenger gets Internet access using a satellite link and an HSDPA link. Reference [13] shows that for a satellite link Mobile IP exhibits better performance than MMP-SCTP (also after the slow start). When there is a handoff between satellite and HSDPA¹¹ we see that Mobile IP does not require any retransmissions, and all packets arrive in order. Very similar results were obtained when the same test was performed using MMP-SCTP [13]. Pareit *et al.* [13] concludes by noting that MMP-SCTP and Mobile IP are able to handle handoffs seamlessly when handoffs can be predicted. In spite of its overhead, MMP-SCTP can be a better choice for a mobility management protocol since it does automatic retransmissions. Pareit *et al.* [13] defer to future work how to decide the optimal instant to make a handoff in order to minimize handoff delay, packet loss and network load.

The possibility of providing Internet access on inter-city trains in California is studied by Kanafani *et al.* in [37]. These researchers propose an architecture for Internet access on trains that

¹⁰Mobile Multi-Path Stream Control Transport Protocol

¹¹Note that [13] only studied handoffs between satellite and HSDPA; however, we expect similar results for other cellular-based systems.

is based on open standard radio technologies, such as IEEE 802.11 and IEEE 802.16, Mobile IP, in-train network components, train to backhaul architecture components, a track-side communication system, a homeland security surveillance system, and command and control centers. In addition this architecture also has a subsystem that would handle handoffs as the train moves from the coverage area of one trackside unit to the next [37]. The train to backhaul architecture component in Kanafani's architecture is analogous to the train access terminal in Fig. 1. The track-side communication system is the access network, while the in-train network is the same as the network shown in Fig. 1.

Most of the papers that we have seen thus far use existing radio technologies, such as IEEE 802.16 [38] or cellular technologies. In [18] Zou *et al.* deviates from most of the previous work, and calls for using IEEE 802.20¹² [39], which is technology under development, to provide broadband Internet access for trains. IEEE 802.20 is chosen because existing 3G technologies do not offer sufficiently high data rates to support many users on a high-speed train. IEEE 802.20, on the other hand, is being designed to support data delivery at high bit rates to vehicles travelling at up to 250 km/h, while using the wireless spectrum efficiently [39]. As in many of the other systems that we have reviewed thus far, Zou *et al.* uses an IEEE 802.11x WLAN on board the train to provide Internet access to passengers. In other words their architecture for the network on-board the train does not deviate significantly from Fig. 1. In order to allow for smooth handoffs between base stations, they call for the train to make two IEEE 802.20 connections to base stations, i.e., the train access terminal in Fig. 1 will make connections to two separate base stations in the access network. However, the train would maintain a single IP address, using Mobile IP, throughout its journey. Furthermore, they argue that since the train's schedule is known, handoff instances should be handled by a Predictive Pre-handover (PPH) algorithm that would pre-compute the routes needed after a handoff. The access node on the train would actively monitor the received signal strength from IEEE 802.20 stations, and it would trigger a handoff whenever the received signal strength from the new station exceeds that of its current base station [18].

With the exception of the system proposing the radio-over-fiber methods for Internet access on trains, hitherto all the systems that we have studied examine communication protocols for providing Internet access. White and Zakharov [40], on the other hand, deal strictly with physical layer issues. They argue that high-altitude platforms, such as airplanes and airships at stratospheric altitudes, are a less costly yet feasible method of providing Internet access to trains. Digital Signal Processing (DSP) algorithms for tracking high-altitude platforms are presented in [40]. The algorithms' purpose is to estimate the direction of arrival (DOA) for signals transmitted from a high altitude platform (HAP) to a train. Some of the methods applied for DOA estimation include Spectral-based [40] and Polynomial-based [40] techniques. An Extended Kalman Filter (EKF) is used to track the train location while beam forming is used on the satellite uplink. Finally the paper shows that EKF can track slow variations in train velocity and account for sudden HAP motion. Null steering (beam steering) is also shown to be advantageous in HAP-train data communications.

Thus far we have covered getting Internet access onto trains; however, we also need to account for the network topology and addresses on the train-based network. Network topology on board the train changes constantly [12], hence there is a need to create a robust management infrastructure that will establish and maintain connectivity on the train while providing logical and IP addressing services [12]. This can be achieved by using the Train Management System

¹²IEEE 802.20 can be seen as the access network technology.

(TMS) architecture, which consists of a network¹³ layer, middleware infrastructure layer, and the user layer. Verstichel *et al.*'s [12] network layer is further subdivided into: the subsystems and components layer (which controls components found throughout the train such as doors, lighting and air conditioning), vehicle layer (which includes all subsystems and services on board of a single car), and the train layer (which results from communications between all the cars on a train). All of these layers are connected by gateways, for example, all of the gateways located in coaches are interconnected using a train-level network, as shown in Fig. 1. The Train Management Scheme uses IP addressing to link the devices in a coach-level network into one network across the entire train. This IP addressing can be done using either IPv4 or IPv6. A summary of the TMS architecture is shown in Fig. 5. Refer to Fig. 1 to review the relationship between the train-level network and the coach-level network.

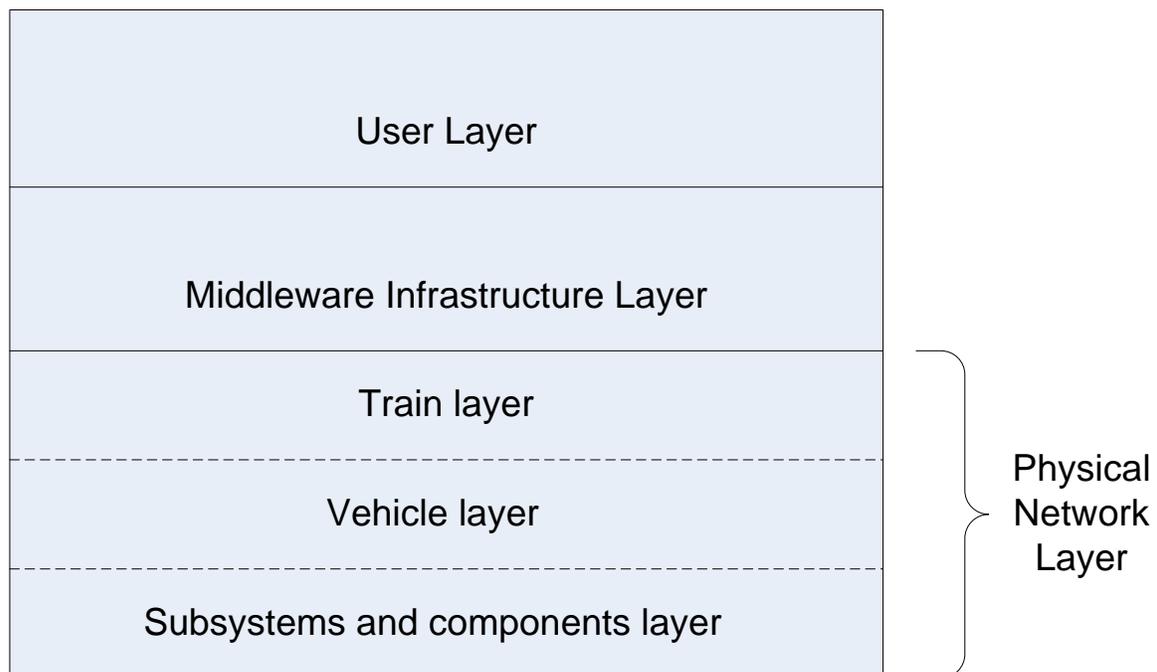


Figure 5: Reference architecture for Train Management System based on [12]

3.2.3 Discussion of Testbed Results and a Feasibility Study

References [41] and [42] begin the transition from the more theoretical to prototypes and deployment. In [41] Sivchenko *et al.* presents simulation results that show that Internet traffic performance on high-speed trains decreases as the number of users increase, which is an expected result. The performance of several existing radio technologies with respect to data rates experienced on fast moving trains is investigated in [42]. Gaspard and Zimmerman [42] evaluate the relationship between throughput as a function of Doppler shift (speed). This investigation was carried out in two phases; in the first stage a channel sounder was used to take channel measurements for different placements of a mobile receiver, while the mobile transmitter was moved along the track. In the next stage different radio technologies were evaluated using a hardware emulation of the channel characteristics. The experiments evaluated how throughput

¹³It should be noted that the network layer in [12] is different from that in the OSI model

would vary for a channel between a trackside transmitter and a receiver on board a train. Experimental results indicate that:

- TCP/IP throughput of a UMTS/FDD downlink does not vary much with receiver input power; however, it is relatively low, i.e., ~0.06–0.35 Mbps.
- At 300 km/h TCP/IP throughput of an IEEE 802.11b link between a trackside transmitter and a receiver on the train varies with receiver input power due to multipath channels. It should be noted that IEEE 802.11b provides high data rates under the measurement conditions. In addition the authors state that one would need several access points along the track to have good coverage.
- The IEEE 802.16 system evaluated in [42] was not suited for high-speed trains since TCP/IP throughput decreased sharply with increasing speed. However, the authors note that the amendments to the IEEE 802.16e standard for mobility should enhance the performance of the IEEE 802.16 system.

Lundberg and Gunningberg [43] study the feasibility of using IEEE 802.11x networking equipment to provide Internet access for a train traveling at 200 km/h between Uppsala and Stockholm. Here they observe that commercial solutions for providing Internet access on trains are available, but note that they are either limited or expensive. Furthermore, they observe that if IEEE 802.11x technology is used, the technology choice will depend on the possible impact of fading and related problems, such as the Doppler effect due to the train's motion [43].

4 Implementation Efforts and Business Models

In the previous section we reviewed the reference architecture and initial concepts underpinning broadband Internet deployment on trains. In this section we look at how those ideas have been implemented in Europe and North America. As we mentioned in Section 1 broadband Internet access is increasingly becoming available on trains in Europe. In Europe the preponderant demand for Internet access is from passengers, while in North America most train traffic is dominated by freight [44]. As a result efforts to carry out communications from trains have evolved in slightly different directions on these two continents due to market forces. We review the implementation efforts in Europe and North America separately, since conclusions drawn from one continent might not necessarily apply to the other. Furthermore, implementation efforts in Europe are much more advanced than those in North America. This section concludes with a look at some of the business models developed, from a technical perspective, to evaluate the viability of broadband Internet access on trains.

4.1 Implementation in Europe

One of the earliest accounts of Internet access on trains comes from the Railway Open System Interconnection Network (ROSIN) project. In 1999 Fabri *et al.* [45] presented a report on a web-based tool deployed to a train to allow maintenance staff to supervise railroad equipment using a GSM connection between the train and an operations center. Aboard the train the railcars were linked into a network using the Train Communication Network (TCN¹⁴) standard. Unfortunately,

¹⁴The TCN specification consists of a train bus and a vehicle bus. The train bus can self-configure itself by connecting a new node (railcar) to the network and dynamically assigning it a new address. The vehicle bus is optimized to handle small packets originating from a large number of devices. The train bus and the vehicle bus are connected through a gateway, which allows for exchange of data between devices in the

reference [45] does not provide any additional details on the bit rates seen during the trial or the network topology.

Ceprani and Schena [46] present implementation details on their Fast Internet for Fast Train Hosts (FIFTH) project. The FIFTH architecture consists of Mobile Train Terminal Prototype (MTTP) and FIFTH Access Network Infrastructure (FANI) modules. The MTTP is composed of a Satellite Access Terminal (SAT), which uses the Ku band to provide satellite access for the train, and the Train User-Local Area Network (TU-LAN) which constitutes the LAN onboard the train. The satellite access terminal is analogous to the train access terminal in Fig. 1, while the Train User-Local Area Network is akin to the rest of the computer network shown in Fig. 1. The antenna for the SAT is adjusted by a Navigation and Tracking Unit during a trip to optimize reception conditions. The TU-LAN consists of a coach LAN (within a train car) and a train LAN (between cars on the train). The TU-LAN is implemented by using Ethernet connections between train cars as well as Ethernet connections and IEEE 802.11 links for passengers to use. Unfortunately, additional details are not available on what bit rates were seen during the trial.

Conti [10] provides a contemporary view of the implementation of Internet access on trains in Europe. In his paper he argues that telecommunications operators have offered Internet access to passengers using GPRS or 3G wireless cards; however, this is not sufficient for most users. Furthermore he states that there is now agreement that Internet access should be provided on board trains using IEEE 802.11 access points within the train; however, there is not much agreement on how to connect moving trains to the Internet backbone. In the United Kingdom GNER trains use a combination of satellite and cellular links to provide a backhaul link from the train. Therefore, the train access terminal in this instance supports both satellite and cellular technologies. The Internet connection is shared with all cars on the train using the train's lighting circuit, this implies that the topology of the computer network on the train is not radically different from that shown in Fig. 1. Unfortunately additional details are not available on how the wired portion of the network aboard the train operates. GNER's system favours satellite access for the backhaul link, but when the train enters a tunnel, the system automatically switches over to GPRS (The technical details of how this switch is accomplished are not clear from [10]; however, it may be assumed that the GPRS signal is brought into the tunnel via a leaky cable, or some similar mechanism). For redundancy purposes the train connects to base stations from two different mobile carriers. In addition up to six parallel cellular phone links are established for redundancy purposes when the train passes through a tunnel. It is worth noting that this same technology is also used by the Swedish train operator, SJ, to provide Internet access [10].

Conti [10] also discusses Southern Trains' efforts to provide Internet access on its trains using WiMax [4]. It is interesting to note that this system does not use any of the enhancements found in IEEE 802.16e, which is designed for mobile access. Instead, this system uses a draft implementation of IEEE 802.16d [9]. Conti [10] adds that T-Mobile and Nomad Digital collaborated on this venture, and that in addition to the pre-WiMax standard, GPRS and 3G technologies are also used for robustness with each train having three GPRS modems for robustness [9]. As of 2005 there were 37 WiMax base stations deployed along the 60 mile train track, with plans to install up to 60 base stations [9]. Each of the base stations was equipped with a 2 Mbps ADSL link to the Internet [9]; each base station in this system could achieve data rates of up to 32 Mbps for both the uplink and the downlink wireless channels [10]. In Southern Trains' implementation the train access terminal consists of a server with support for WiMax and

same railcar, or in two different train cars. The TCN can also be linked to the Internet by means of a radio link between the train and a ground station.

GPRS technologies. The architecture of the in-train network is akin to that shown in Fig. 1, with passengers connecting to the in-train network using an IEEE 802.11b link [9]. Finally, the access network in this case uses WiMax and GPRS [9], while the aggregation network uses ADSL [9].

Apart from WiMax and GPRS technologies, satellite technologies may also be used for Internet access. For example, elsewhere in Europe, Thalys [10] uses a bi-directional satellite link operating in the Ku-band to support link speeds of up to 2 Mbps, i.e., the train access terminal only supports satellite links. The downside of relying on satellite links is that operational costs are probably higher than for links that rely on either WiMax or 3G technologies [10].

Echensperger [17] discusses work done by T-Mobile in Germany to bring Internet access to intercity trains. He discusses the Railnet effort, which aims to provide WLAN access on board trains while also providing a broadband radio connection between the train and the land side. The Railnet system uses a Central Train Unit to control traffic and store on-board content, several antennas to maintain the train to base station link, an IEEE 802.11 network to link the rail cars into a train level network, and IEEE 802.11 access points on-board the train for passenger access. The on-board network for the Railnet effort is very similar to that shown in Fig. 1, except that there are no wired links between the railcars. Instead this time we have IEEE 802.11x links between the cars. The train access terminal in this case supports T-Mobile's access technology¹⁵. Since T-Mobile (the service provider) owns its network, and also provides service on board the train, there is not much of a distinction between the access and aggregation networks in this case. It is worth noting that Flash-OFDM has also been evaluated in the course of the Railnet effort, and its throughput has been found to be nearly independent of velocity [17].

4.2 Implementation in North America

As previously mentioned rail transportation in North America and Europe have very different characteristics. Consequently broadband Internet deployment to trains on those continents has evolved differently. In fact, it could even be argued that these deployments are in their infancy in North America. However, there are some efforts underway for North America. For example, Conti [10] points out that PointShot Wireless has worked on initial deployments with Canada's VIA Rail and California's Altamont Commuter Express and Capitol Corridor operators.

A lot of the work coming from North America is experimental, given the lack of widespread Internet access on board trains. One example of some experimental work comes from the University of Nebraska, where Hempel *et al.* [16] presents work done on a wireless testbed for IEEE 802.11 deployed along a train track. In this testbed IEEE 802.11 access points were placed along the tracks with line of sight paths to neighboring access points. This arrangement allowed for seamless IEEE 802.11 coverage along the tracks. IEEE 802.11a channels were used to provide backhaul links between the testbed access points, while IEEE 802.11b was used to provide wireless Internet connectivity to the train car used in the tests. Results from the testbed showed that IEEE 802.11b could support data rates of up to 11 Mbps; however, IEEE 802.11b was also subject to interference from passing trains. Additional test results showed that train velocity does not appear to have a significant effect on the throughput experienced by the node on board the train. The conclusion from this paper is that while it is feasible to deploy IEEE 802.11 along the train track, IEEE 802.11 has a limited coverage area; therefore, such a deployment would be expensive [16]. In addition we have already seen from [42] that the TCP/IP

¹⁵Unfortunately, technical details on the access technology are not available in [17].

throughput of an IEEE 802.11b link varies with receiver input power. Hence, IEEE 802.11x is not suitable for providing Internet access to trains.

More recently Nomad Digital collaborated with the Utah Transit Authority (UTA) and Wasatch Electric to provide a wireless broadband connection on a commuter line between Ogden and Salt Lake City. In this case the access network consists of WiMax radios from Redline Communications. On board the train passengers get Internet access from a free Wi-Fi connection [11]. The on-board network for this rail deployment is very similar to that shown in Fig. 1, while in this instance the train access terminal supports WiMax. Unlike in any of the examples seen thus far, the aggregation network in this instance is composed of fiber optic links, some of which run trackside [11].

Most of the work we have reviewed in this paper has discussed providing Internet access to passengers on a train. However, a train operator might also like to collect operational data from its trains. Edwards *et al.* [2] discuss just such a scheme that allows for controlling and monitoring various sensors and supervision modules on a freight train. This scheme uses IEEE 802.11b for intra-train communications to allow for braking, coupling and uncoupling, etc. This scheme uses a Controller Area Network (CAN) bus to collect data from sensors on board the train. The data is then coupled with GPS information and reported to a web server via a CDMA-based transmitter. In this case the train access terminal is a 1xRTT radio, whereas the links between the cars are IEEE 802.11b links; unlike the wired links shown in Fig. 1.

4.3 Business Models for Internet Service on Trains

As we have seen in previous sections broadband Internet access is increasingly being deployed to trains. However, for us to see more widespread deployments, train operators would have to be convinced of the business advantages of such a deployment. In this subsection we present different business models for paying for Internet service on trains.

One of the earliest business models developed studied deploying Internet access to inter-city trains in California [37]. In developing this model, the authors say that the provision of Internet access on trains would likely lead to an increase in ridership on the inter-city trains. The train operators on the other hand could collect revenue from this service either by applying “per use or time charges, subscription fees,” or negotiating an arrangement with a third party to pay for the service through advertising, or sponsorship, or an increase in ridership [37]. In the case of California trains, the authors present two business models for providing Internet access:

- Option 1 is a conservative model that uses satellite and cellular networks for backhaul, with an IEEE 802.11 access network on the train. This option has a low operational cost with low bandwidth and a high operational cost with high bandwidth, but it generally results in low revenue for the train operator. This option is aimed at capturing mobile Internet users on trains in a conservative manner.
- Option 2, uses WiMax for backhaul access with an on-board Wi-Fi network, but it has a high initial cost (due to the cost of deploying WiMax antennas) with low operational costs. Kanafani *et al.* [37] state that this model should result in high revenue for the train operator, and that it should help capture mobile Internet users as the market grows.

The next two business models were developed for use in Europe. Using data from Belgian railways, Lannoo *et al.* [34] present business models that investigate the possibilities and economic viability of providing Internet access on trains. Recall that these researchers are part of the same group that proposed the FAMOUS architecture. As in previous work, they argue that

broadband Internet access on trains can be provided by using an in-train network, and a network between the train and the service provider for Internet access. For the backhaul network trains can use cellular networking technologies such as GPRS/UMTS/HSDPA, or wireless networking technologies such as Wi-Fi, WiMax, Flash-OFDM, or even a satellite networking standard, such as DVB-S/DVB-S2/DVB-RCS. These backhaul networks can be classified as either incumbent networks, for example GPRS/UMTS/HSDPA, or dedicated networks, for example, WiMax or Flash-OFDM, or satellite networks. With incumbent networks the goal would be to provide Internet access on trains without making a major capital expenditure. The business model presented in [34] considers using incumbent networks until their capacity requirements are exceeded, then one can roll out a dedicated network. Satellite networks would only be used as gap fillers, i.e., in areas where the other networking standards do not provide adequate coverage, just as we saw in [10]. The analysis carried out in [34] assumes revenue schemes where:

1. every passenger pays for Internet service, or
2. only first class passengers get free Internet access, while all other passengers pay.

Their analysis also includes the capital expenses required for deploying Internet service, as well as the operational costs required to maintain service. The model then presents results to show that train operators would realize a net profit if only first class passengers get free Internet access. Lannoo *et al.* [34] conclude by noting that using a combination of technologies is the best way to provide broadband Internet access to trains, and that in the particular case of Belgian railways it would be better to use a mix of WiMax and UMTS for Internet access [34].

More recently Riihimaki *et al.* [35] have studied Finnish railroads to determine the feasibility of deploying broadband Internet to trains. They argue that revenue from providing Internet service to train customers may come from the following sources:

1. An increase in passenger volume, if a train operator offers free Internet access for passengers
2. An increase in the number of first class passengers, if first class passengers get free Internet access.
3. Reduced personnel costs, if passengers who buy their tickets online get free on-board Internet access.
4. Direct revenue, if train tickets and data connections are sold separately

From the standpoint of the train operator Internet access on trains could allow for more efficient train operations, e.g., allowing real-time traffic control, or more efficient staff who can verify passenger tickets in real-time.

Hitherto, we have focussed on Internet access to passengers, Riihimaki *et al.* state that train operators shipping freight could use a broadband Internet connection to allow their customers to perform accurate cargo monitoring. In the case of the Finnish railroads, it is argued that the cost of building a network for Internet access from trains can be spread out over a period of time if the network is built in two or more phases, for example by using GPRS or Flash-OFDM in the first phase, and then using mobile WiMax in the second phase. Furthermore, in the case of WiMax they show that the average revenue collected per user, and the cell range of the WiMax network are the most critical parameters influencing this technology's viability for Internet access on trains [35]. For example, their analysis is based on an estimated WiMax cell size of 5 km. However, if this cell size is decreased by 10% then it becomes unprofitable to provide Internet access using WiMax [35].

Given that most of the train traffic in North America is freight traffic [44], possibly the best avenue for getting broadband Internet access on trains would be to forge some kind of partnership between the train companies and telecommunications companies. If the train operators can see a reduction in their insurance payments by allowing freight customers to gain visibility into their shipments or other gains in efficiency, then the long-term viability of broadband Internet on trains may be achieved in North America. In the case of the United States, Amtrak passengers can also benefit from a deployment of broadband Internet access to trains, and perhaps even more people can be lured to riding trains in the United States resulting in lower greenhouse emissions.

5 Conclusion

The availability of broadband Internet access on trains should prove to be a revenue source for operators. Previous studies from the United Kingdom show that train companies can attract more users if Wi-Fi access is made available [1]. In this paper we have presented some of the initial approaches, current technologies, and future ideas, such as IEEE 802.20 and radio-over-fiber, related to Internet access on trains. We have also provided an account of implementation efforts for broadband Internet access on trains in Europe and North America. These efforts, particularly from Europe, show that broadband Internet access on trains is realizable. Furthermore, business models, developed to test the viability of Internet access on trains, show that broadband Internet access on trains is best realized by using a combination of access technologies. However, efficient operation requires proper system design. North America does not share the same rail traffic characteristics as Europe [44], and so broadband Internet access on North American trains is not as readily available. In North America broadband Internet access on trains may be used for collecting operational data from trains, as well as freight monitoring. Future work could be to develop a business model for broadband Internet access on North American trains that takes into account the fact that North American rail traffic is dominated by freight. A good business model might serve to accelerate the deployment of broadband Internet access in North America.

Acknowledgment

The author would like to thank Ms. Yewande Lewis for reading and commenting on a previous version of this paper.

References

- [1] BBC News. (2004, May 20) Wi-Fi May Tempt Travellers. News. BBC News. London, United Kingdom. [Online]. Available: <http://news.bbc.co.uk/2/hi/technology/3729583.stm>
- [2] M. C. Edwards *et al.*, “Improving Freight Rail Safety with on-board Monitoring and Control Systems,” in *Proceedings of the 2005 ASME/IEEE Joint Rail Conference*, Pueblo, CO, USA, Mar. 2005, pp. 117–122.
- [3] BBC News. (2004, July 6) Rail Users Get Wi-Fi Net Access. News. BBC News. London, United Kingdom. [Online]. Available: http://news.bbc.co.uk/2/hi/uk_news/england/3868585.stm
- [4] B. Wilson. (2005, Oct. 26) Rail Internet Access Picks Up Speed. News. BBC News. London, United Kingdom. [Online]. Available: <http://news.bbc.co.uk/2/hi/business/4363196.stm>
- [5] B. Lannoo *et al.*, “Radio-over-fiber-based Solution to Provide Broadband Internet Access to Train Passengers,” *IEEE Communications Magazine*, vol. 45, no. 2, pp. 56–62, Feb. 2007.
- [6] I. Beeby, “Demystifying Wireless Communications for Trains,” Presented at the BWCS Train Communication Systems 2006, London, UK, June 2006.
- [7] P. A. Laplante and F. C. Woolsey, “IEEE 1473: An Open-Source Communications Protocol for Railway Vehicles,” *IT Professional*, vol. 5, no. 6, pp. 12–16, November/December 2003.
- [8] I. Beeby, “The Future for Terrestrial Wireless Services for the next Five Years: Myths and Realities for WiFi on Trains,” Presented at the BWCS Train Communication Systems 2007, London, UK, June 2007.
- [9] P. Judge. (2005, Apr. 3) 100 mph WiMax hits the rails to Brighton. News. TechWorld. United Kingdom. [Online]. Available: <http://www.techworld.com/mobility/features/index.cfm?FeatureID=1351>
- [10] J. P. Conti, “Hot Spots on Rails,” *Communications Engineer*, vol. 3, no. 5, pp. 18–21, Oct./Nov. 2005.
- [11] Nomad Digital. (2008, May 21) U.S. First for Nomad Digital: WiFi provided free for all rail passengers. Press Release. Nomad Digital. Newcastle, United Kingdom. [Online]. Available: http://www.uknomad.com/news_details19.html
- [12] S. Verstichel, K. Lamont, F. De Turck, B. Dhoedt, P. Demeester, and F. Vermeulen, “On the Design of a Train Communication Management Platform,” in *Symposium on Communications and Vehicular Technology*, Liège, Belgium, Nov. 2006, pp. 29–34.
- [13] D. Pareit *et al.*, “QoS-enabled Internet-on-train network architecture: inter-working by MMP-SCTP versus MIP,” in *7th International Conference on ITS Telecommunications (ITST '07)*, Sophia Antipolis, France, June 2007, pp. 1–6.
- [14] F. Van Quickenborne *et al.*, “Managing Ethernet Aggregation Networks for Fast Moving Users,” *IEEE Communications Magazine*, vol. 44, no. 10, pp. 78–85, Oct. 2006.
- [15] F. De Greve *et al.*, “FAMOUS: A Network Architecture for Delivering Multimedia Services to FAst MOving USers,” *Wireless Personal Communications*, vol. 33, no. 3-4, pp. 281–304, 2005.
- [16] M. Hempel *et al.*, “A Wireless Test Bed for Mobile 802.11 and Beyond,” in *IWCMC '06: Proceedings of the 2006 International Conference on Wireless Communications and Mobile Computing*. Vancouver, BC, Canada: ACM, 2006, pp. 1003–1008.
- [17] H. Echensperger, “Railnet: High-Speed Internet on High-Speed Trains,” Presented at the IET Seminar: Broadband on Trains, London, United Kingdom, Feb. 2007.

- [18] F. Zou, X. Jiang, and Z. Lin, "IEEE 802.20 Based Broadband Railroad Digital Network - The Infrastructure for M-Commerce on the Train," in *The Fourth International Conference on Electronic Business - Shaping Business Strategy in a Networked World (ICEB)*, Beijing, China, 2004, pp. 771–776.
- [19] F. De Greve *et al.*, "Towards Ethernet-Based Wireless Mesh Networks for Fast Moving Users," in *EUROMICRO '06: Proceedings of the 32nd EUROMICRO Conference on Software Engineering and Advanced Applications*. Dubrovnik, Croatia: IEEE Computer Society, Aug. 2006, pp. 387–397.
- [20] F. De Greve *et al.*, "Design of Wireless Mesh Networks for Aggregating Traffic of Fast Moving Users," in *MobiWac '06: Proceedings of the 4th ACM International Workshop on Mobility Management and Wireless Access*. Terromolinos, Spain: ACM Press, Oct. 2006, pp. 35–44.
- [21] F. Van Quickenborne *et al.*, "Tunnel Set-up Mechanisms in Ethernet Networks for Fast Moving Users," in *11th International Telecommunications Network Strategy and Planning Symposium (NETWORKS 2004)*, Vienna, Austria, June 2004, pp. 303–308.
- [22] F. De Greve *et al.*, "Aggregation Network Design for Offering Multimedia Services to Fast Moving Users," in *Quality of Service in Multiservice IP Networks: Third International Workshop, (QoS-IP 2005)*, M. A. Marsan, G. Bianchi, M. Listanti, and M. Meo, Eds., vol. LNCS 3375/2005. Catania, Italy: Springer-Verlag New York, Inc., Feb. 2005, pp. 235–248.
- [23] -----, "Cost-effective Ethernet Routing Schemes for Dynamic Environments," in *GLOBECOM'05: IEEE Global Telecommunications Conference*, vol. 2. St. Louis, MO, USA: IEEE, Nov. 2005, pp. 1023–1028.
- [24] -----, "Rapidly Recovering Ethernet Networks for Delivering Broadband Services on the Train," in *LCN'05: The 30th IEEE Conference on Local Computer Networks*. Sydney, Australia: IEEE Computer Society, Nov. 2005, pp. 294–302.
- [25] B. Lannoo, D. Colle, M. Pickavet, and P. Demeester, "Extension of the Optical Switching Architecture to Implement the Moveable Cell Concept," Presented at the ECOC 2005: Proceedings 31st European Conference on Optical Communication, vol. 4, Glasgow, United Kingdom, Sep. 2005, paper Th 1.4.3, pp. 807–808.
- [26] L. M. Correia and R. Prasad, "An Overview of Wireless Broadband Communications," *IEEE Communications Magazine*, vol. 35, no. 1, pp. 28–33, Jan. 1997.
- [27] C. D. Gavrilovich, "Broadband Communication on the Highways of Tomorrow," *IEEE Communications Magazine*, vol. 39, no. 4, pp. 146–154, Apr. 2001.
- [28] K. D. Lin and J. F. Chang, "Communications and Entertainment Onboard a High-speed Public Transport System," *IEEE Wireless Communications Magazine*, vol. 9, no. 1, pp. 84–89, Feb. 2002.
- [29] T. Van Leeuwen *et al.*, "Broadband Wireless Communication in Vehicles," in *FITCE 2003: 42nd European Telecommunications Congress*, Berlin, Germany, Sep. 2003, pp. 77–82.
- [30] F. De Greve *et al.*, "Evaluation of a Tunnel Set-up Mechanism in QoS-aware Ethernet Access Networks," in *LANMAN 2004: The 13th IEEE Workshop on Local and Metropolitan Area Networks*, San Francisco, CA, USA, April 2004, pp. 247–252.
- [31] F. Van Quickenborne *et al.*, "Optimization Models for Designing Aggregation Networks to Support Fast Moving Users," in *Proceedings 1st Int. Workshop of the EURO-NGI Network of Excellence on Wireless Systems and Mobility in Next Generation Internet*, G. Kotsis and O. Spaniol, Eds., vol. LNCS 3427. Dagstuhl, Germany: Springer, June 2005, pp. 66–81.

- [32] F. De Greve *et al.*, “A New Carrier Grade Aggregation Network Model for Delivering Broadband Services to Fast Moving Users,” *International Journal of Communication Systems*, vol. 20, no. 3, pp. 335–364, Mar. 2007.
- [33] B. Jooris *et al.*, “Mobile Communication and Service Continuity in a Train Scenario,” Presented at the Proceedings of the 12th Symposium on Communications and Vehicular Technology in the BENELUX, Enschede, Netherlands, Nov. 2005.
- [34] B. Lannoo *et al.*, “Business Model for Broadband Internet on the Train,” in *Proceedings of the 46th Federation of Telecommunications Engineers of the European Community Congress (FITCE 2007)*, Warsaw, Poland, Aug. 2007, pp. 60–66.
- [35] V. Riihimaki *et al.*, “Techno-economical Inspection of High-speed Internet Connection for Trains,” *IET Intelligent Transport Systems*, vol. 2, no. 1, pp. 27–37, Mar. 2008.
- [36] G. Bianchi *et al.*, “Internet Access on Fast Trains: 802.11-based on-board wireless distribution network alternatives,” in *12th IST Mobile & Wireless Communications Summit*, Aveiro, Portugal, June 2003, pp. 15–18.
- [37] A. Kanafani *et al.*, “California Trains Connected,” University of California - Berkeley, Tech. Report UCB-ITS-PRR-2006-4, Apr. 2006.
- [38] *Air Interface for Fixed Broadband Wireless Access Systems*, IEEE Std. 802.16, 2004.
- [39] *Draft Standard for Mobile Broadband Wireless Access (MBWA)*, IEEE Draft Standard 802.20-D1, 2006.
- [40] G. P. White and Y. V. Zakharov, “Data Communications to Trains From High-Altitude Platforms,” *IEEE Transactions on Vehicular Technology*, vol. 56, no. 4, pp. 2253–2266, July 2007.
- [41] D. Sivchenko *et al.*, “Internet Traffic Performance in High Speed Trains,” in *HET-NETs '04: Second International Working Conference: Performance Modelling and Evaluation of Heterogeneous Networks*, Ilkley, United Kingdom, July 2004, pp. 26–31.
- [42] I. Gaspard and G. Zimmermann, “Investigations for Broadband Internet within High-speed Trains,” *Advances in Radio Science*, vol. 3, no. 13, pp. 247–252, May 2005.
- [43] D. Lundberg and P. Gunningberg, “Feasibility Study of WLAN Technology for the Uppsala - Stockholm Commuter Train,” Department of Information Technology, Uppsala University, Tech. Rep., June 2004.
- [44] J.-P. Rodrigue, C. Comtois, and B. Slack, *The Geography of Transport Systems*. New York, NY USA: Routledge, 2006, ch. 3. Transportation Modes, p. 284.
- [45] A. Fabri, T. Nieva, and P. Umiliacchi, “Use of the Internet for Remote Train Monitoring and Control: the ROSIN Project,” Presented at the Rail Technology Conference, London, United Kingdom, September 1999.
- [46] F. Ceprani and V. Schena, “FIFTH Project Solutions Demonstrating New Satellite Broadband Communication System for High Speed Train,” in *VTC 2004: Proceedings of the 59th IEEE Vehicular Technology Conference, Spring*, vol. 5, Milan, Italy, May 2004, pp. 2831–2835.



Technical Report

**Status Update: A Unified Architecture for SensorNet
with Multiple Owners: Supplement to Advance
SensorNet Technologies to Monitor
Trusted Corridors**

University of Kansas
Telecommunication Technology Center
V.S. Frost, G.J. Minden, J.B. Evans,
L. Searl and D.T. Fokum

EDS
T. Terrell, L. Sackman, M. Gatewood, J. Spector,
S. Hill, and J. Strand

ITTC-FY2009-TR-41420-10
August 2008

Project Sponsor:
Oak Ridge National Laboratory (ORNL)
Award Number 4000043403

Abstract

This effort is aimed at monitoring cargo movements along a trusted corridor, e.g., rail facilities, in association with an integrated data-oriented methodology to increase efficiency and security. This goal is being achieved by performing research and deployment of an associated testbed focused on rail transportation issues. The results of this effort will lay the foundation for enhancing the ability of the private sector to efficiently embed security that provides business value such as safety, faster transport and reduced theft while supporting law enforcement and national security. In the end, the benefit of the combination of real-time sensor data with trade data exchange information will be demonstrated through field tests on a deployed rail testbed.

Table of Contents

Abstract.....	i
Table of Contents.....	ii
List of Figures.....	ii
List of Tables.....	ii
1.0 Introduction.....	1
2.0 Status on Technology Proof of Concept and Integration of the SmartPort Trade Data Exchange and Transportation Security SensorNet Technologies.....	2
3.0 Status of the Development of Transportation Security SensorNet (TSSN) Technologies.....	3
4.0 Status of System Architecture, Modeling, and Optimization.....	9
5.0 Status of Communications System Evaluation.....	11
6.0 Status RFID Technology Evaluation and Development.....	13
7.0 Associated Efforts.....	13
8.0 Project Timeline.....	15
9.0 References.....	16
10.0 Appendixes.....	16

List of Figures

Figure 1: Field Trial.....	2
Figure 2: Selected Proof of Concept Technologies.....	3
Figure 3: TSSN Implementation.....	4
Figure 4: Mobile Rail Network.....	5
Figure 5: Virtual Network Operations Center.....	5
Figure 6: Example Train Configuration (red square indicates deployed sensor).....	10
Figure 7: Relative System Cost vs Number of Sensors.....	11
Figure 8 (from ⁷): Cost per Function Trend.....	11
Figure 9: Cost/Capability Trade-off for Communications Devices.....	12
Figure 10: Future Trends in Sensor Technology (From [4]).....	13
Figure 11.....	14
Figure 12: Project Timeline.....	15

List of Tables

Table 1: ACE SOA Status.....	6
Table 2: TSSN Phase 1 Status.....	7
Table 3: TSSN Phase 2 Status.....	8

1.0 Introduction

This project is demonstrating the tracking and monitoring technologies needed to establish a trusted corridor for international and domestic cargo movements along a path including inter-modal facilities. The results of this effort will lay the foundation for enhancing the ability of the private sector to efficiently embed security that provides business value, e.g., faster transport and reduced theft, while supporting law enforcement and national security.

Exports from Asia have increased, creating bottlenecks at key US ports. A Kansas City (KC) group, known as SmartPort, recognized the strategic position of KC and has actively worked to increase shipments through the KC area. SmartPort is developing a US export capability and will have the only Mexican Customs clearance capability that is not at the border. This project is aimed at improving the efficiency and security of these trade lanes by combining monitoring, real-time tracking, and associated sensor information with trade data exchange (TDE) information. The focus is on technology development to address transportation issues, and validation of the concepts via the deployment of a testbed.

KC SmartPort, through the Mid America Regional Council (MARC), is fostering the development of several trade lane projects. SmartPort/MARC supported EDS to execute the International Corridor Integration Project (ICIP), which demonstrated a reduction in international transport shipping time from KC to Mexico from 10-14 days to 3 days. SmartPort/MARC through its Intelligent Transportation Integration Project is supporting EDS to develop a Trade Data Exchange (TDE) that captures commercial clearance and other data. The ORNL SensorNet initiative is aimed at developing the technology, standards, and technical requirements for an integrated national warning and alert system to provide an incident discovery, awareness, and response capability, addressing local, regional, and national needs. Thus the ORNL SensorNet provides the basis for obtaining the real-time tracking and associated sensor information. ITTC/KU has been focused on creating technologies that will allow SensorNet to interact in an environment composed of multiple enterprises, owners, and operators of the infrastructure, including sensors and TDE. ITTC/KU research has addressed assured and controlled access to SensorNet assets, implying a focus on security and management mechanisms; and archives and information dissemination, including interfaces/schemas. This effort is focused on creating an interoperable TDE and SensorNet. A cost-effective communications system to facilitate continuous monitoring of containers and communications is also under study as is the role of evolving Radio Frequency Identification (RFID) technologies. In the end, the benefit of the combination of SensorNet with TDE information will be demonstrated through field tests on a deployed testbed.

This effort is being executed by a team from EDS and ITTC/KU. There have been close interactions between the EDS and ITTC/KU teams. There have been weekly conference calls to coordinate activities. The project conference calls have involved Steve Peterson and Randy Walker from ORNL.

The next section addresses the integration of a distributed sensor system with the TDE, along with the status of the proof of concept field trial. EDS is leading the TDE and field trial efforts. The status of development of technologies effort to facilitate the participation of multiple organizations is described next. This aspect of the effort addresses access/control/security mechanisms. The status of the system architecture, modeling, and optimization will then be addressed. The communications system required for monitoring will be addressed in the next section followed by the status on the associated RFID

efforts. We conclude by discussing the relationship of this effort to several aligned activities; a timeline for the remainder of the project is then presented.

2.0 Status on Technology Proof of Concept and Integration of the SmartPort Trade Data Exchange and Transportation Security SensorNet Technologies

Here an integrated Transportation Security SensorNet (TSSN) and Trade Data Exchange environment is under development where the prototype will be evaluated in a field trial. A requirement of the prototype will be to create commercial value for a SmartPort stakeholder using the TSSN and TDE technologies.

Though an extensive series of meetings and interactions, including trips to locations at the US/Mexico boarder (Laredo, Texas), a SmartPort stakeholder has been identified and agreed to participate in the field trial. This stakeholder will be providing access to their facilities in Kansas City and Mexico; they will also allocate personnel time to support the field trial at no cost to this project. Three field trials are planned; short haul tests as well as two trials in Mexico.

Specific use cases were identified in consultation with the SmartPort stakeholder. Interactions with the SmartPort stakeholder combined with the use cases resulted in the selection of devices to be employed for sensing, processing, and communications. The use cases center around monitoring and tracking containers with the goals of proving that a container breach did not occur during the stakeholder's custody and providing time and location of a container intrusion to enable the stakeholder's response and reduce successful intrusions. Figure 1 shows the test environment.

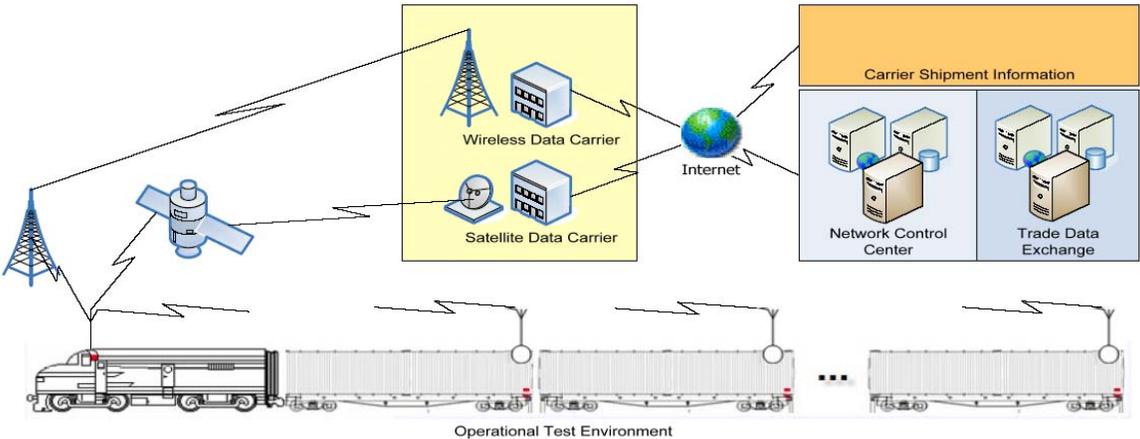


Figure 1: Field Trial

Figure 2 shows the selected proof of concept technologies. The associated equipment has been acquired and is being integrated into a complete system. This includes servers to host the TDE at EDS and the seals, tags, reader, vehicle mounted TSSN collector (laptop), and a virtual network operations center (VNOC) functionality at ITTC/KU. Experiments have been conducted with the Hi-G-Tek seals,

tags, reader, and software developed to integrate them into this system. Initial communications and interactions have been established between the TDE at EDS and the VNOC in the TSSN at ITTC/KU.

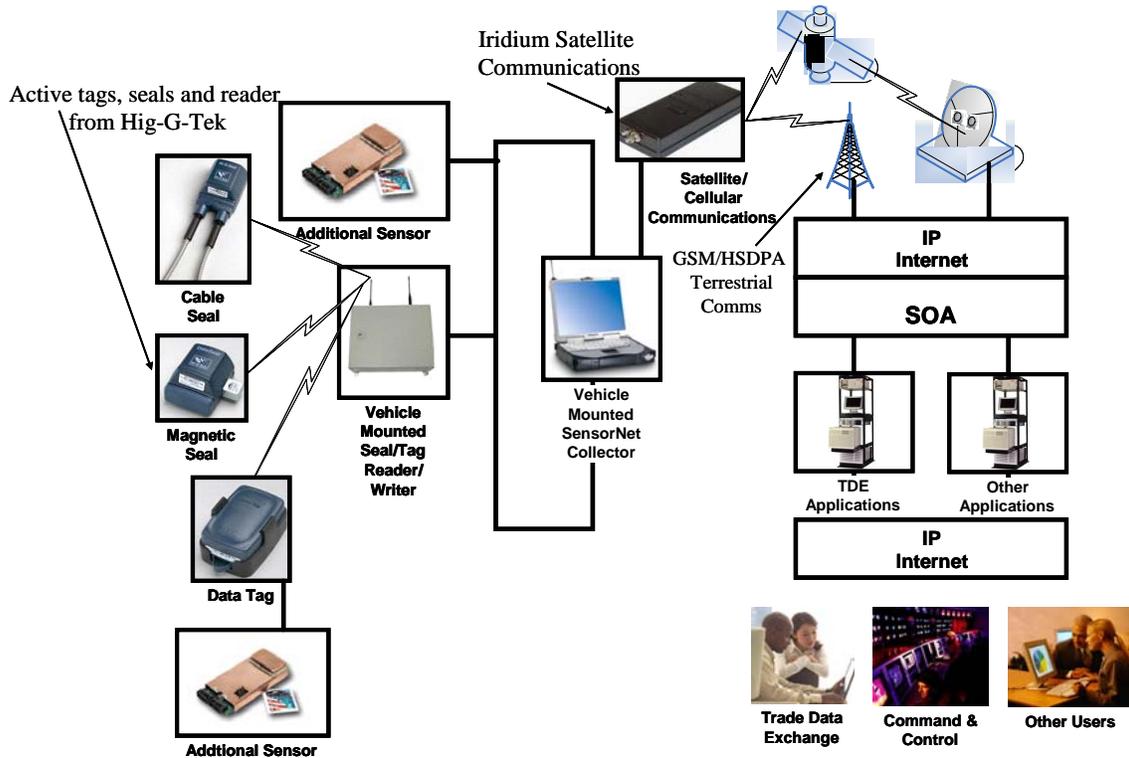


Figure 2: Selected Proof of Concept Technologies

Integration with the TDE has been started. A test plan to describe how we will unit test, field test and conduct integrated testing is under development.

3.0 Status of the Development of Transportation Security SensorNet (TSSN) Technologies

The TSSN technologies are built upon the results of previous development efforts. The TSSN technologies use the ACE SOA (Ambient Computing Environment) (Service Oriented Architecture) which is the fourth generation ACE implementation for providing distributed computing, media, and sensing services to service consumers (clients) in a dispersed environment. ACE SOA is the infrastructure providing message and data communication, confidentiality, authenticity, and permissions plus service discovery within an enterprise and between enterprises. Also provided is a framework for developing new services and clients of services. ACE concepts were used in developing an initial prototype of a SensorNet with multiple owners [1]. This effort moves beyond the previous work; the ACE SOA is a reimplement of the original ideas of ACE but utilizing current technology and widely accepted open Web Service specifications and publicly available implementations which are suitable for Sensor Networks. Some of the Web Service specifications in use are SOAP, the WS-X specifications, and UDDIv3. The ACE SOA infrastructure allows Web Service based clients and services of one or more enterprises to interact using the following features:

- Provide means for service to publish its URL location and Web Service Interface for discovery by clients.
- Allow clients to discover service's URL location and Web Service Interface.
- Provide a secure communication channel between clients and services.
- Provide mechanism for clients to subscribe to service 'events' or 'alarms'.
- Authenticate a client to a service.
- Provide fine grain authorization of a client's use of a service.
- Provide a framework for development of new clients and services.
- Establish a trust relationship between enterprises

A detailed discussion of the ACE SOA is in Appendix A [2].

The TSSN using the ACE SOA is being implemented in three phases. The first phase will be used in the field trials described above.

Phase1 – Simple service messages based on OGC specifications (used in trials)

Phase2 – Use full OGC specification interface messages.

Phase3 – Use lessons learned from Phase1 and 2 to make improvements

The TSSN implementation is composed of SOA Infrastructure for TSSN, the VNOC, and the Mobile Rail Network (MRN) as shown in Figure 3. Some detail of the VNOC and MRN as well as the interworking with the TDE is shown in Figure 4 and Figure 5. The status of SOA infrastructure for TSSN as well as each phase is given below in Tables 1, 2, and 3.

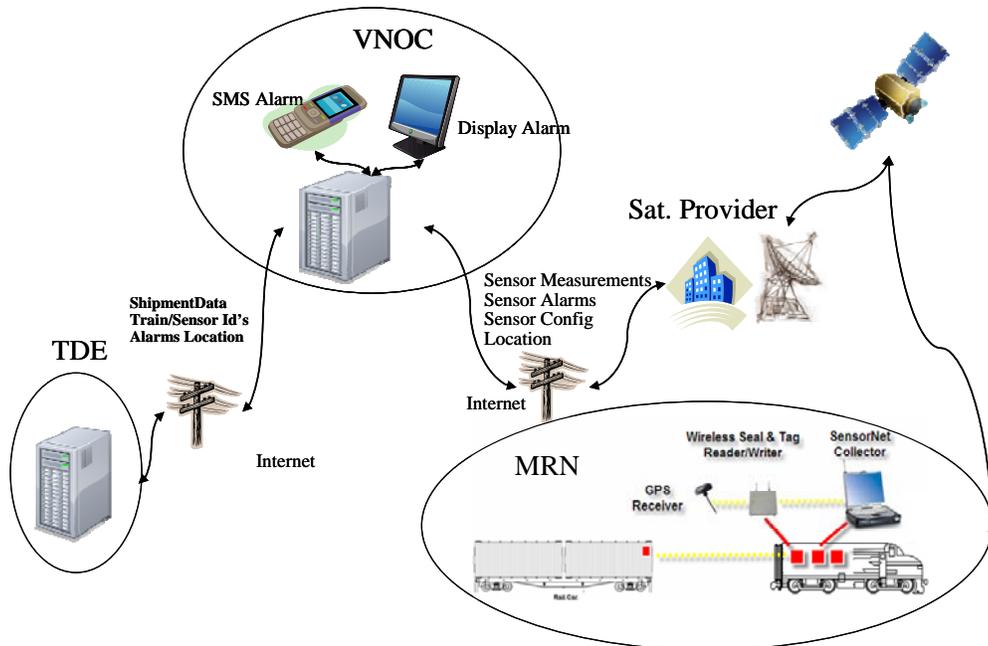


Figure 3: TSSN Implementation

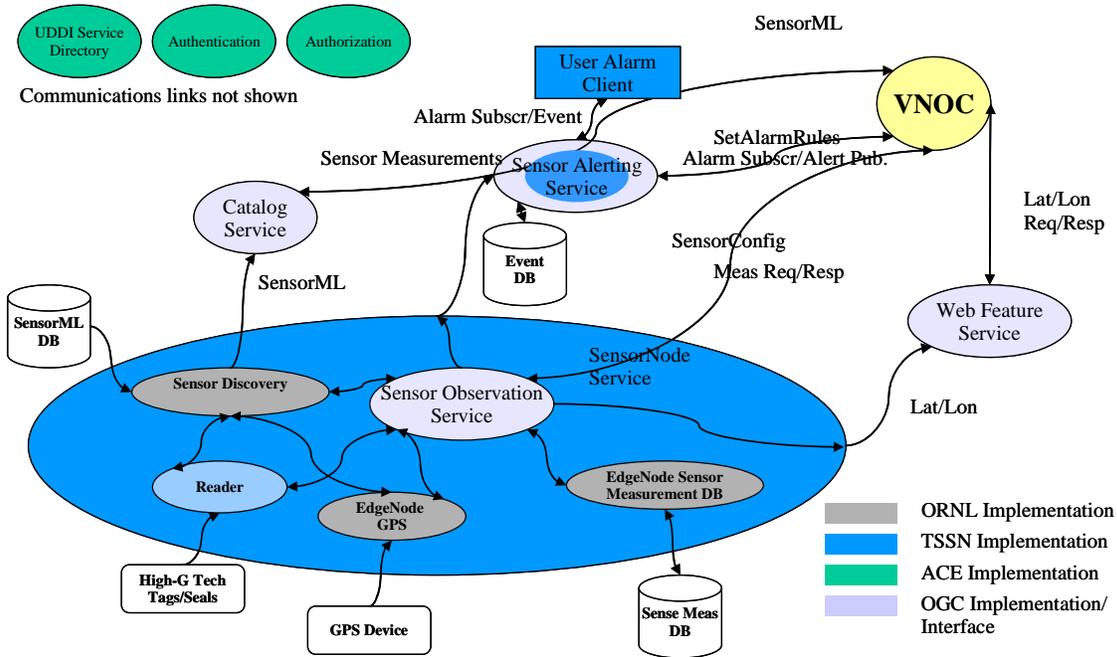


Figure 4: Mobile Rail Network

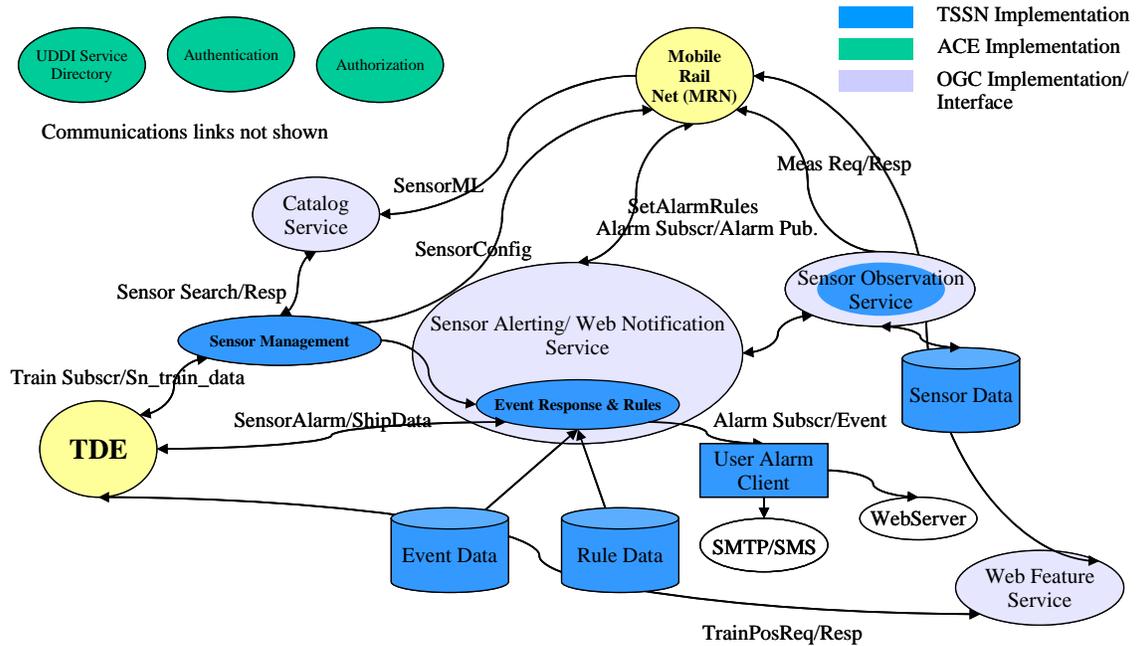


Figure 5: Virtual Network Operations Center

Functionality	% Complete, Tasks
Transport	100% - SOAP/HTTP - Apache Axis2 implementation (v1.4)
Confidential Data	100% - WS-Security, HTTPS - Axis2 implementation (Rampart)
Remote Exception	100% - SOAP Fault - Axis2 has extendable exception mechanism
Server Alert/Alarm	100% - WS-Eventing - Implemented mechanism for stand alone clients to receive events. - TODO: Switch over to WS-BasicNotification
Authentication, Client-Server	100% - WS-Security - Axis2 implementation (Rampart)
Intra-Enterprise Authentication Service (signed token)	80% - Axis2 implementation (Rampart) - Use Standard Token Service (STS) - TODO: decide on token and signing mechanism/type - Decision may be impacted by load data rate of Iridium
Inter-Enterprise Authentication Trust	20% Complete - WS-Federation/WS-Trust specifies mechanism (Axis2 Rampart) - Use Standard Token Service (STS) - TODO: Verify that implementation provides required functionality
Intra-Enterprise Authorization Service	0% - WS-Authorization has not been written - Can use xACML as language
TDE Integration	50% - Testing with local TDE using EDS provided WSDL - TODO: use EDS located TDE service.
Intra-Enterprise Service Discovery	90% - UDDI v3, OpenUDDI Implementation (v0.9.8) - Implemented common service code for automatic publishing - Implemented common client code for simplified service discovery. - TODO: complete enterprise service to clean up stale UDDI info.
Inter-Enterprise Service Discovery	50% - UDDI v3, OpenUDDI Implementation replication - Each enterprise has public UDDI for replication with other enterprise public UDDIs - TODO: Need enterprise service to publish public services to public UDDI.
Auditing/Monitoring	50% - Implemented message logging module for Axis2 - Implemented GUI for message monitoring - TODO: Evaluate current utility of module and GUI and make improvements.

Table 1: ACE SOA Status

Functionality	% Complete, Tasks
MRN Communication Service	50% <ul style="list-style-type: none"> - Basics of setting up network connection over Iridium and GSM complete - Can measure GSM signal strength for connection switch over decision - TODO: write SOA service code
MRN Sensor Node Service	85% <ul style="list-style-type: none"> - Can process all service operations (start, stop, GetCapabilities, etc) - Can generate alert events based on simulated sensor events - TODO: complete last HGT AVL Reader commands
MRN Alarm Processing Service	70% <ul style="list-style-type: none"> - Can receive Alerts from Sensor Node - Can do simple if/then/else event processing and publish alarms to subscribers - TODO: finish Complex Event Processing code using Esper.
MRN Alarm Reporting Client	100% <ul style="list-style-type: none"> - Subscribes to MRN Alarm Processing Service for Alarms - Uses simple text output.
NOC Sensor Management Service	100% <ul style="list-style-type: none"> - Accepts TDE start operation - Sends start operation to MRN Sensor Node
NOC Alarm Processor Service	70% <ul style="list-style-type: none"> - Subscribes to MRN Alarm Processor for Alarm events - Can receive Alarms from MRN Alarm Processor - Can do simple if/then/else event processing and publish alarms to subscribers - TODO finish Complex Event Processing code using Esper.
NOC Alarm Reporting Service	100% <ul style="list-style-type: none"> - Subscribes to NOC Alarm Processor for Alarm Events - Receives Alarm events and notifies users by SMS message and/or Email based in information in user notification database

Table 2: TSSN Phase 1 Status

Functionality	% Complete, Tasks
MRN Alarm Reporting Client	0% - TODO: Change to using a web browser interface
MRN Sensor Node Service	0% - TODO: Use full OGC service interface (SOS, SAS, FaultReport)
MRN Alarm Processing Service	0% - TODO: Use full OGC SAS Alert and FaultReport - TODO: Develop rules for Complex Event Processing based on GPS, Cargo Info and Sensor Readings
NOC Sensor Management Service	0% - TODO: Use full OGC FaultReport
NOC Alarm Processing Service	0% - TODO: Use full OGC FaultReport - TODO: Develop rules for Complex Event Processing Develop rules for Complex Event Processing
NOC Alarm Reporting Service	0% - TODO: Publish user notifications to subscribed clients. This is really just for the NOC Alarm Reporting Client
NOC Alarm Reporting Client	0% - TODO: Similar to MRN Alarm Reporting Client but also subscribes to NOC Alarm Reporting Service

Table 3: TSSN Phase 2 Status

4.0 Status of System Architecture, Modeling, and Optimization

This task is focused on developing models of the system that can be used to articulate trade-offs and enable optimization. To develop the understanding required for creating the models we first studied the taxonomy of sensor network architectures, see Appendix B [3]. In [3] several proposed architectures for sensor networks are reviewed. We observed that there is a lack of an over-arching sensor network architecture. In [3] we present some of the issues associated with existing sensor network architectures followed by a discussion of several specific cases, including the one for multi-owner environment associated with this effort [1]. We also classify these architectures in terms of function and compositional elements. We also highlight each architecture's key attributes in order to identify their commonalities. In making our arguments we refer to the concept of invariants, which are components of a system that cannot be changed without losing backward compatibility. Our results show that these architectures share several invariants.

Also as part of developing appropriate models, we conducted a survey on methods for broadband internet access on trains, see Appendix C. Here we studied approaches for providing broadband Internet access to trains and examined some of the factors that hinder their deployment. This survey exposes some of the basic concepts for providing broadband Internet access and then reviews associated network architectures. The review of network architectures shows that we can subdivide networks for providing broadband Internet access on trains into the train-based network, the access network — for connecting the train to the service provider(s), — and the aggregation network for collecting user packets generated in the access network for transmission to the Internet. Furthermore, our review shows that the current trend is to provide Internet access to passengers on trains using IEEE 802.11x; however, a clear method for how to connect trains to the global Internet has yet to emerge. A summary of implementation efforts in Europe and North America serves to highlight some of the schemes that have been used thus far to connect trains to the Internet.

There are many ways to deploy TSSN technology. At one extreme every container could be assigned sensors and reach back communications capability, at another extreme only “valuable” containers could be assigned a sensor and a low cost radio for communications to a single collector and reach back communications system, and at another extreme there is no reach back communications capability on the train and track side readers are deployed. There are trade-offs in all of these cases. The model developed here is aimed at addressing these system trade-offs.

There are also several success metrics to consider, e.g.,

- System operational cost. This metric is computed per trip, and it consists of the system's false alarm cost, the cost of deploying the sensors, repeaters and readers, and the backhaul communications device, as well as the cost of reporting events in a deployed cargo monitoring system. The costs of missing an event at a given load (container) as well as the costs of a communications failure at a sensor are also components of this metric.
- Weighted sum of probability of sensor detection. This metric is computed by adding the probabilities of detection at each of the loads weighted by each load's value. Ideally the goal would be to maximize

this metric, so that it is as close to 1 as much as possible. It should be observed that as this metric is increased the cost of a missed detection is reduced leading to a decrease in the system operational cost.

- Weighted sum of probability of false alarm. This metric is computed by adding the probabilities of false alarm at each load weighted by the loads' values. Here the goal would be to minimize this metric, so that it is as close to 0 as much as possible.
- Sensor network lifetime. It is assumed that all the sensors in the network would be battery powered. Consequently we would strive to maximize the sensor network lifetime, while keeping the probability of detection above some threshold value.
- Time taken for event to be successfully received at decision point. With this metric, lower values would always be desirable. This metric would be computed on a per-load basis, and we assume that if a load does not have a sensor, then an event can be detected at that load only when the train arrives at its destination. If, on the other hand, the load carries a sensor, then this metric consists of the time required to detect the event plus the time taken to report an event.

The model under development will enable the study of system trade-offs with respect to these metrics. As an illustrative example suppose that the containers are in fixed positions on the train as shown in Figure 6, and sensors are attached to selected loads (containers). Here the collector of sensor information and the backhaul communications device is located in the locomotive. In this case we evaluate how the system operational cost varies as sensors are added to the containers in order of value.

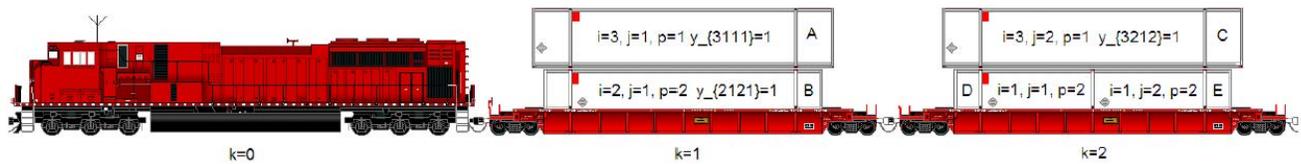


Figure 6: Example Train Configuration (red square indicates deployed sensor)

We have developed an initial model and objective function that enables us to address this question. The model when completed will provide a way to address complex system trade-offs. Under our initial simplified assumptions (details of this analysis beyond the scope of this report) the system operational cost as sensors are added to the containers in order of value is given in Figure 7. Note deploying zero sensors means a system with no protection and has the highest relative system cost; this reflects the high cost of missing an event.

Further work is needed on the model, objective function, and obtaining realistic model parameters. Then the framework needs to be applied to study system trade-offs. A contribution of the effort will be this complex system model.

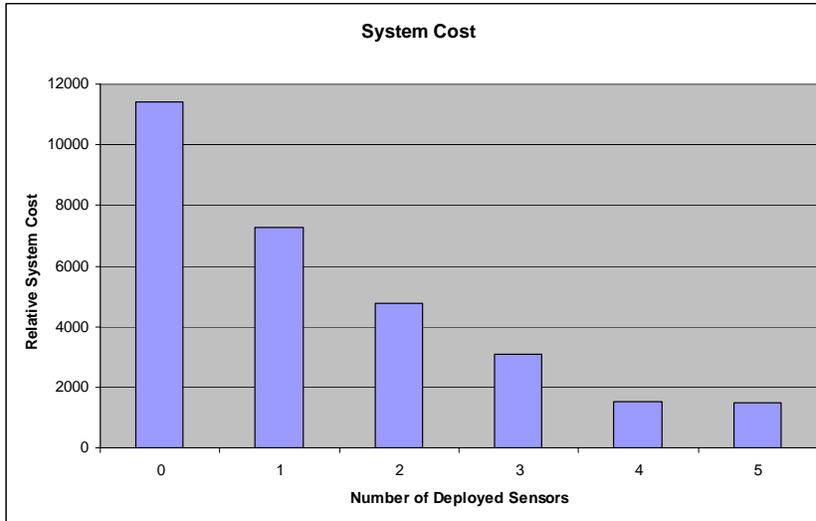


Figure 7: Relative System Cost vs Number of Sensors. Relative system cost includes cost of missing an event, deployment costs, cost of false alarms and communications costs.

5.0 Status of Communications System Evaluation

Providing visibility, accountability, efficiency, and security requires the coordinated application of sensing, communications, and the integration of information. There are several trends that need to be recognized.

a) Increased processing capabilities

The cost of memory and computing has been continually decreasing as a result of Moore's law. This trend is expected to continue, enabling new uses of embedded intelligence like those proposed here. Figure 8 (from¹) shows the cost per function trend and forecasts predict an additional two orders of magnitude in cost/performance by 2010.

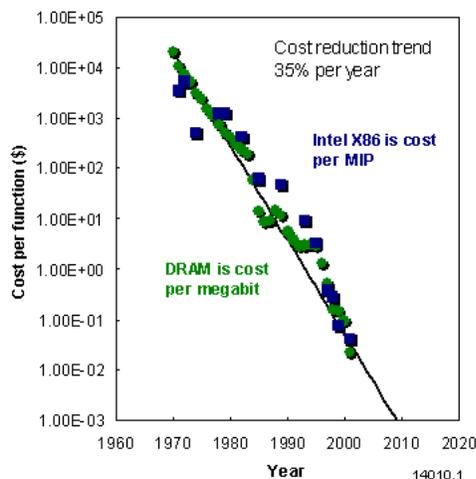


Figure 8 (from⁷): Cost per Function Trend

¹ <http://www.icknowledge.com/economics/productcosts2.html>

b) Increased communications capabilities

The communications architecture is composed of: 1) local communications, e.g., using two-way systems like the SunSpot² or ZigBee³, 2) communications between groups of container, e.g., using WiFi or Vehicle Infrastructure Integration-VII (see below for details), and 3) reach-back to backbone networks, e.g., using, satellite, cellular technology or VII in some circumstances. Note that communications “holes” are a property of cellular technology that can result in problems when applied in the transportation domain, e.g., loss or mistaken tracking of containers. All of these elements of wireless networking technology continue to advance, resulting in increased capability at lower costs.

The Vehicle Infrastructure Integration (VII) initiative⁴ aims to endow vehicles with the ability to communicate to other vehicles and surface transportation infrastructure to promote public safety. The system is intended to warn drivers of impending problems, as well as provide real-time data for the transportation infrastructure to increase efficiency. Other applications are also envisioned, ranging from in-vehicle entertainment to measuring weather and road conditions from information transmitted and collected using the VII⁵. The VII also specifies the use of Dedicated Short Range Communication (DSRC) technology operating at a frequency band of 75 MHz centered at about 5.9 GHz; this band has been allocated for this application by the FCC. The VII is in its early stages of development, however 1) it is clear that VII can play a role here and 2) the VII has not yet been applied in a freight security environment.

Figure 9 shows the conceptual tradeoff between communications costs and capabilities. The trend is that the cost of these devices will continually decrease, enabling new roles for each technology. This trend will enable wider spread of two-way communications device.

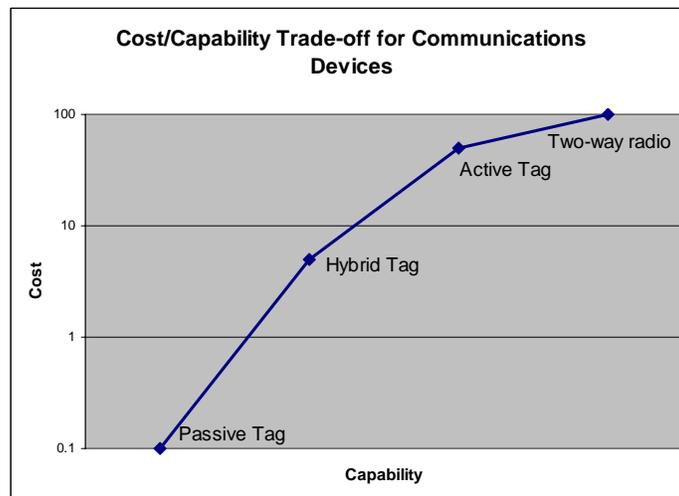


Figure 9: Cost/Capability Trade-off for Communications Devices

² <http://www.sunspotworld.com/>

³ <http://www.zigbee.org/en/index.asp>

⁴ Vehicle Infrastructure Integration (VII): VII Architecture and Functional Requirements-Version 1.1, FHWA, ITS Joint Program Office, US Department of Transportation, July 20th, 2005.

<http://www.ral.ucar.edu/projects/vii/docs/VIIArchandFuncRequirements.pdf>

⁵ VII Weather Applications Workshop Boulder, Colorado, June 21, 2006.

http://www.ral.ucar.edu/projects/vii/PresWrkShop1/VII_Goals_Motivation_%20Petty.ppt

c) Improved sensing technologies

Sensors will continue to decrease in size and cost, increase in integrated signal processing, applied in multisensor configurations, and be more closely integrated with wireless communications capabilities [4]. These are summarized in Figure 10 [4]. Research will continue radio technologies for TSSN.

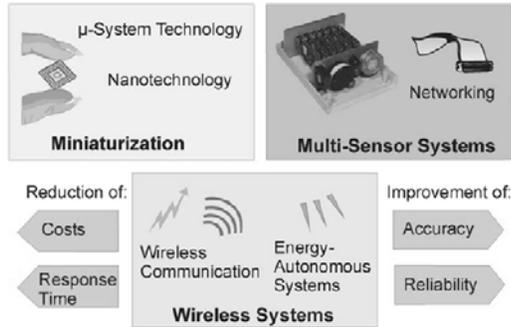


Figure 10: Future Trends in Sensor Technology (From [4])

6.0 Status RFID Technology Evaluation and Development

Knowledge of the identification and location of containers and other mobile elements of the system is a key component of the trusted corridor concept. RFID technologies will play a major role in providing this knowledge. RFID technology has the potential to determine fine scale location of elements, e.g., containers. To be effectively deployed RFID technology, seals and/or readers must account for multiple operating frequencies. The technology enabling operation on-metal, i.e., on a container, of passive RFIDs at multiple frequencies (covering both Europe and the USA) has been developed at ITTC/KU and is described in Appendix D [5]. Systems have begun to appear which provide passive RFID based location-detection see [6]. The combination of the new ITTC/KU on-metal RFID tag technology and the Mojix system offers a potential solution to the identification and location of containers issues in intermodal facility, especially a train yard and train-to-truck transportation.

The combination of the new ITTC/KU on-metal RFID tag technology and the Mojix system was deployed and tested in a warehouse environment. While this initial testing focused on the suitability of the system on a MES (manufacturing and execution system, i.e., an assembly line) and for scanning entering and exiting a dock door, the results of this testing lead to conclusions concerning applicability in an intermodal environment. Analysis of the results of those tests is under way. Additional experiments are planned.

7.0 Associated Efforts

The current effort is aimed at monitoring cargo movements along a trusted corridor in association with an integrated data-oriented methodology. This goal is being achieved by performing research and a deployment of an associated testbed focused on rail transportation issues. A rail partner has been identified and has agreed to participate in a field trial in the US and Mexico; the technology concepts

developed will be validated via deployment of this testbed. The field trial is currently planned for fall of 2008. This effort is integrating TSSN with the TDE information for correlation between documents and sensed environment, and integrating real-time tracking information for correlation between documents and tracking data. The effort is standards-based, leverages the Service Oriented Architecture (SOA), appropriate OGC efforts and web technologies. A unique partnership with SmartPort/MARC and EDS (EDS is a subcontractor to KU/ITTC) has resulted from this effort. The Kansas City SmartPort organization has recognized the strategic transportation position of Kansas City. Kansas City SmartPort, through the Mid-America Regional Council (MARC), is encouraging the development of several trade lane projects.

KC SmartPort/MARC is supporting the Cross Town Improvement Project (CTIP). The goal of CTIP is to build a database application that supports basic rail and truck interchange information needs in Kansas City. This capability will allow users to view all equipment available for return to the dray's origin terminal. The aim is to enable the seamless, efficient and safe movement of legitimate intermodal freight between facilities in and around economically vital metropolitan areas. CTIP is also standards based and uses a web-based SOA approach.

US-DoT is supporting Electronic Freight Management (EFM) initiative activities in association with KC SmartPort/MARC in the Kansas City area. The EFM initiative is an effort aimed at improving data and message transmissions between supply chain partners. Their goal is to provide a mechanism for sharing supply chain freight information that is simpler, cheaper and more efficient than traditional EDI, enabling supply chain partners to access the information, and make it easier to customize the flow of information between partners. EFM is also standards based and uses a web-based SOA approach.

KC SmartPort has recognized the potential of leveraging TSSN, CTIP and EFM to positively impact the freight movement supply chain through logistics transactions (CTIP and EFM) and field sensing (TSSN). TSSN is the critical element with respect to security and possible visibility by appropriate government agencies. The combined vision is shown in Figure 11.

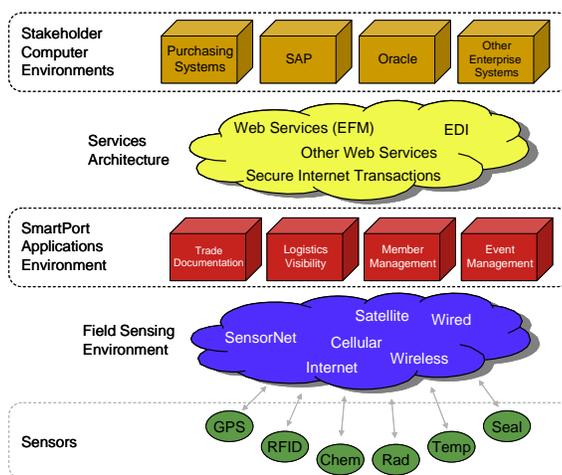


Figure 11

The combination of these efforts has significant potential. Recognizing this potential, in June 2008, KC SmartPort started to coordinate monthly meetings for the groups involved in TSSN, CTIP and

EFM. The goal is to create a common, open environment with low entry barriers to enable broader access by stakeholders while contributing a venue to commercialization. The KU/ITTC and EDS teams are supporting the interactions between these efforts to realize the vision shown in Figure 11. This activity will ensure issues associated with security and possible visibility by appropriate government agencies are considered.

8.0 Project Timeline

Figure 12 is the current project time line. The field trial is target for completion by then end of 2008. The efforts associated with the system modeling, communications, and RFID are planned to be completed by about April 2009 and a report describing these activities delivered by mid-June 2009. Activities associated with SmartPort, EFM, and CTIP will continue until June 2010.

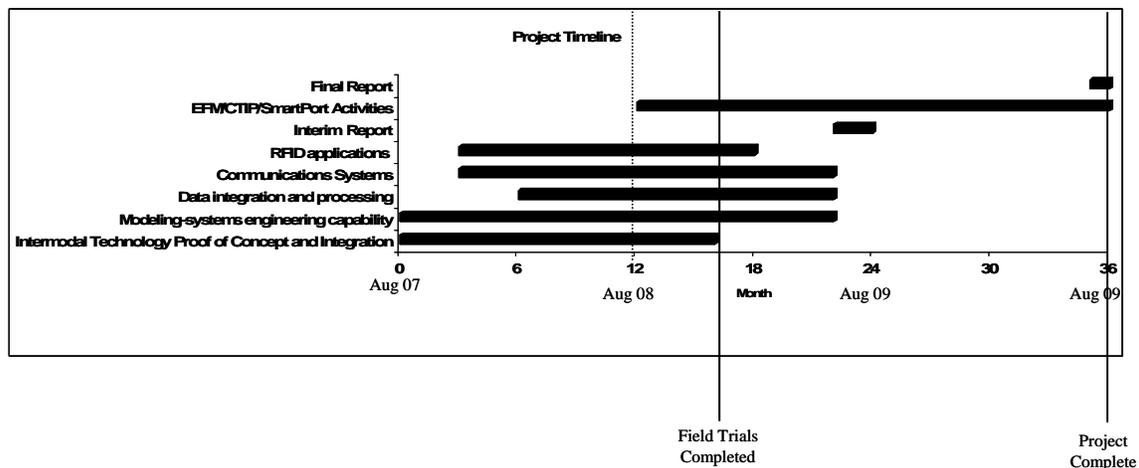


Figure 12: Project Timeline

9.0 References

- [1] Gary Minden, Victor Frost, David Petr, Douglas Niehaus, Ed Komp, Daniel Fokum, Pradeepkumar Mani, Andrew Boie, Satyasree Muralidharan, and James Stevens, "Phase One Report: A Unified Architecture for SensorNet with Multiple Owners," Technical Report ITTC-FY2008-TR-41420-06; December 2007.
- [2] Leon S. Searl, "Service Oriented Architecture for Sensor Networks Based on the Ambient Computing Environment," Technical Report, ITTC-FY2008-TR-41420-07, February 2008.
- [3] D.T. Fokum, V.S. Frost, P. Mani, G.J. Minden, J.B. Evans, and S. Muralidharan, "A Taxonomy of Sensor Network Architectures," ITTC-FY2009-TR-41420-08 July 2008.
- [4] Kanoun, O. and H.-R. Tränkler, *Sensor technology advances and future trends*. IEEE Transactions on Instrumentation and Measurement, 2004. **53**(6): p. 1497-1501.
- [5] Supreetha Aroor and Daniel D. Deavours, "A Dual-Resonant Microstrip-Based UHF RFID "Cargo" Tag," Proc. International Microwave Symposium, June 15-20, 2008, Atlanta, GA, USA.
- [6] <http://www.mojix.com/>

10.0 Appendixes

Appendix A. Leon S. Searl, "Service Oriented Architecture for Sensor Networks Based on the Ambient Computing Environment," Technical Report, ITTC-FY2008-TR-41420-07, February 2008

Appendix B. D.T. Fokum, V.S. Frost, P. Mani, G.J. Minden, J.B. Evans, and S. Muralidharan, "A Taxonomy of Sensor Network Architectures," ITTC-FY2009-TR-41420-08 July 2008.

Appendix C. Daniel T. Fokum, Victor S. Frost, "A Survey on Methods for Broadband Internet Access on Trains" ITTC-FY2009-TR-41420-xx July 2008.

Appendix D. Supreetha Aroor and Daniel D. Deavours, "A Dual-Resonant Microstrip-Based UHF RFID "Cargo" Tag," Proc. International Microwave Symposium, June 15-20, 2008, Atlanta, GA, USA



Technical Report

Experiences from a Transportation Security Sensor Network Field Trial

Daniel T. Fokum, Victor S. Frost, Daniel DePardo,
Martin Kuehnhausen, Angela N. Oguna,
Leon S. Searl, Edward Komp, Matthew Zeets,
Daniel D. Deavours, Joseph B. Evans,
and Gary J. Minden

ITTC-FY2009-TR-41420-11

June 2009

Project Sponsor:
Oak Ridge National Laboratory
Award Number 4000043403

CONTENTS

I	Introduction	4
II	System Architecture	6
II-A	Trade Data Exchange	7
II-B	Virtual Network Operations Center	8
II-C	Mobile Rail Network	9
II-C1	Mobile Rail Network Hardware	9
II-C2	Mobile Rail Network Software	11
III	Experiments	12
III-A	Road Test with Trucks	12
III-B	Short-haul Rail Trial	14
IV	Postprocessing of Experimental Data	15
V	Results	17
V-A	VNOC to MRN to VNOC Interaction	18
V-B	Elapsed Time from Alert Generation to AlarmReporting Service	19
V-C	End-to-end Time from Event Occurrence to Decision Maker Notification	21
V-D	TDE to VNOC to TDE Interaction	23
V-E	VNOC to TDE to VNOC Interaction	25
V-F	Summary of Time Statistics	26
V-G	Messages by Schema Element	26
V-H	Message Sizes	27
V-I	Intercommand and Interalarm Times	29
V-J	HSDPA Signal Strength	29
VI	Impact on System Modeling	30
VII	Refinements Based on Preliminary Results	32
VIII	Conclusion	33
	Acknowledgments	33

References

33

LIST OF FIGURES

1	Transportation Security Sensor Network (TSSN) Architecture	7
2	Virtual Network Operations Center Architecture	8
3	TSSN Collector Node Hardware Configuration	9
4	Container Seal	11
5	Mobile Rail Network Collector Node Architecture	11
6	Partial Map of Road Test with Event Annotations	13
7	Short-haul Rail Trial Configuration	14
8	Partial Screen Shot of e-mail Message Sent During Trial	15
9	LogParser Framework Showing Message Couples and Transmit/receive Pairs	16
10	Request/response and Network Times from VNOC → MRN → VNOC	19
11	Processing Times at MRN	19
12	Sequence Diagram with Messages Involved in Decision Maker Notification	20
13	Elapsed Time from Alert Generation to VNOC AlarmReporting Service	21
14	Elapsed Time from Event Occurrence to Alert Generation	22
15	Time Taken to Deliver SMS Messages for All Carriers	23
16	Request/response and Network Times from TDE → VNOC → TDE	24
17	Processing Times at VNOC	24
18	Request/response and Network Times from VNOC → TDE → VNOC	25
19	Processing Times at TDE	25
20	Component Interactions in the TSSN	27
21	Intercommand and Interalarm Times at MRN	29
22	HSDPA Signal Strength versus Time	30
23	HSDPA Signal Strength and Geographical Location	31

LIST OF TABLES

I	Summary of Time Taken to Deliver SMS Messages	23
II	Summary of Time Statistics	26
III	Number of Messages Generated by Schema Element	28
IV	Summary of Message Size Statistics	28

Abstract

Cargo shipments are subject to hijack, theft, or tampering. Furthermore, cargo shipments are at risk of being used to transport contraband, potentially resulting in huge fines to shippers. We seek to mitigate these risks through development of a Transportation Security Sensor Network (TSSN) based on open software systems and Service Oriented Architecture (SOA) principles. The TSSN is composed of three geographically distributed components: the Mobile Rail Network (MRN), Virtual Network Operations Center (VNOC), and the Trade Data Exchange (TDE). Using commercial off-the-shelf (COTS) sensors, the TSSN is able to detect events and report those relevant to appropriate decision makers. Two experiments have been conducted to assess the TSSN's suitability for monitoring rail-borne cargo. Log files were collected from these experiments and postprocessed. In this paper we present empirical results on the interaction between various components of the TSSN. These results show that the TSSN can be used to monitor rail-borne cargo. We also discuss some of the research issues that must be addressed before the TSSN can be deployed.

Index Terms

Service oriented architecture, Mobile Rail Network, Trade Data Exchange, Virtual Network Operations Center

I. INTRODUCTION

In 2006 the FBI estimated that cargo theft cost the US economy between 15 and 30 billion dollars per year [1]. Cargo theft affects originators, shippers, and receivers as follows: originators need a reliable supply chain in order to stay afloat, but cargo thefts adversely affect the reliability of the supply chain (A receiver's ability to receive goods in a timely manner affects the originator.). Shippers, on the other hand, hold liability and insurance costs for shipments thus, they would like to maintain low costs due to cargo theft. Finally, receivers are impacted by out-of-stock and scheduling issues due to cargo theft. Most non-bulk cargo travels in shipping containers. Container transport is characterized by complex interactions between shipping companies, industries, and liability regimes [2]. Stakeholders (originators, shippers, and receivers) are looking for a higher degree of visibility, accountability, efficiency, and security in complex container transport chains. Deficiencies in the container transport chain expose the system to attacks such as the Trojan horse (the commandeering of a legitimate trading identity to ship an illegitimate or dangerous consignment), hijack, or the theft of goods. Insufficiencies in these areas can be overcome by creating secure trade lanes (or trusted corridors), especially at intermodal points, for example, at rail/truck transitions. Research and development is underway to realize the vision of trusted corridors.

The work described here focuses on: advanced communications, networking, and information technology applied to creating trusted corridors. The objective of the research is to provide the basis needed to improve the efficiency and security of trade lanes by combining real-time tracking and associated sensor information with shipment information. One crucial research question that must be answered in order to attain this objective is how to create technologies that will allow continuous monitoring of containers by leveraging communications networks, sensors as well as trade and logistics data within an environment composed of multiple enterprises, owners, and operators of the infrastructure. The resulting technologies must be open and easy to use, enabling small and medium sized enterprises (SMEs) to obtain the associated economic and security benefits.

To achieve improved efficiency and security of trade lanes, we have developed a Transportation Security Sensor Network (TSSN), based on Service Oriented Architecture (SOA) [3] principles, for monitoring the integrity of rail-borne cargo shipments. The TSSN is composed of a Trade Data Exchange (TDE) [4], Virtual Network Operations Center (VNOC), and Mobile Rail Network (MRN). The functions of each of these components are discussed in greater detail in Section II. The TSSN detects events and reports those important to decision makers using commodity networks. For the TSSN to be deployed we need to understand the timeliness of the system; however, we do not know *a priori* how the TSSN would perform due to the unknown execution time of SOA-based programs ([5] and [6]), unpredictable packet latency on commodity networks, and the slow and potentially unreliable nature of SMS (Short Message Service) [7] for alarm notification. Thus, we have carried out two experiments to characterize the TSSN system, particularly the end-to-end time between event occurrence and decision maker notification using SMS. The data collected from these experiments will be used in models to investigate system trade-offs and the design of communications systems and networks for monitoring rail-borne cargo.

In this paper we present a high-level description of our cargo monitoring system and experimental results documenting the interactions between various components of the TSSN. These results indicate that decision makers can be notified of events on the train in a timely manner using the TSSN. The rest of this paper is laid out as follows: In Section II we present a description of the TSSN system architecture including the components. Section II also discusses the hardware configuration used in the MRN. In Section III we discuss two experiments conducted to assess the suitability of the TSSN system for cargo monitoring. Section IV discusses the framework used to postprocess the log files from our experiments. Section V presents empirical results showing the interaction between various components of the TSSN. In Section VI we discuss how the empirical results can be used in a model to determine optimal or near-optimal sensor placement. Section VII discusses some refinements to the TSSN architecture based

on preliminary results. Finally, we provide concluding remarks in Section VIII.

II. SYSTEM ARCHITECTURE

To achieve the objectives presented in Section I we have built a system called the Transportation Security Sensor Network (TSSN). The SOA and web services used in the TSSN enable the integration of different systems from multiple participating partners. Moreover, the use of SOA and web services enable data to be entered once and used many times. Using commercial off-the-shelf (COTS) sensors, the TSSN is able to detect events and report those relevant to shippers and other decision makers as alarms. Furthermore, the TSSN supports multiple methods for notifying decision makers of alarms.

The TSSN uses open source implementations of Web service specification standards such as Apache Axis2 [8] and OpenUDDI [9]. Axis2 is an implementation of the Simple Object Access Protocol (SOAP) [8], where SOAP is used in Web services to exchange structured information between a service provider and a requester [10]. Universal Description Discovery Integration (UDDI), on the other hand, provides a service directory and allows a “standard-based approach to locate and invoke a service, and manage metadata relating to that service [10].” Support for multiple owners and users is done through use of WS-Authorization, WS-Trust, and WS-Federation. Our current TSSN prototype uses sensors and readers from Hi-G-Tek [11]. Moreover, the TSSN supports terrestrial communication technologies such as HSDPA (High-Speed Downlink Packet Access) [12] and satellite communication technologies such as Iridium [13]. The use of HSDPA and Iridium allows decision makers to be notified of alarms through SMS (Short Message Service) and/or e-mail messages. There are cost and performance benefits to using both HSDPA and Iridium, including the following: it is cheaper and faster to send messages over an HSDPA link versus an Iridium link; on the other hand, a satellite link is needed as an access technology in those parts of the countryside where an HSDPA connection is unavailable.

Since the TSSN system is currently a prototype, there is a need to gather log files that will allow for system debugging as well as to capture metrics that can be used to evaluate system performance. Logging is currently done at the MRN, VNOC, and TDE using Apache log4j [14]. Log4j enables “logging at runtime without modifying the application binary [14].”

The TSSN system is composed of three major geographically distributed components: the Trade Data Exchange (TDE), Virtual Network Operations Center (VNOC), and the Mobile Rail Network (MRN), as shown in Fig. 1. Each of these components is presented in greater detail in the following subsections.

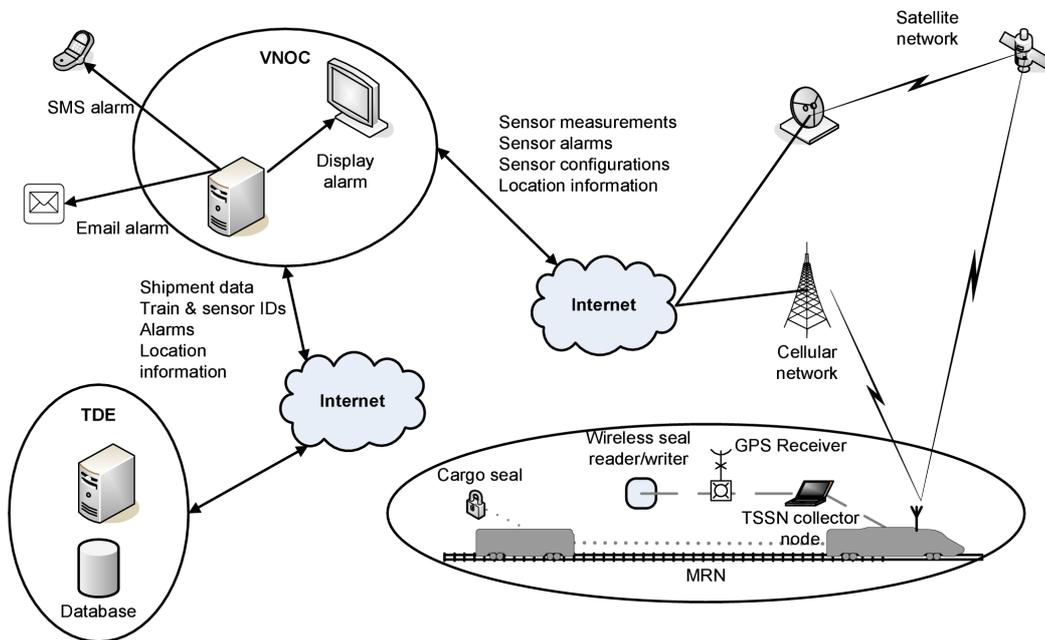


Fig. 1. Transportation Security Sensor Network (TSSN) Architecture

A. Trade Data Exchange

The Trade Data Exchange (TDE) contains shipping data and it interconnects commercial, regulatory and security stakeholders. The TDE is based on a “technology-neutral, standards-based, service-oriented architecture [4].” The TDE is hosted on a server that is geographically separated from the VNOG, and it responds to queries from the VNOG. The TDE also stores alarm messages sent by the VNOG. Finally, the TDE sends startMonitoring, stopMonitoring, and getLocation messages to the VNOG.

In addition to the functions mentioned above, the TDE will monitor the progress of shipment and other logistics information. The TDE captures commercial and clearance data including: the shipping list, bill of lading, commercial invoice, Certificate of Origin (for example, NAFTA Letter), and shipper’s export declaration. It also validates and verifies data to ensure accuracy, consistency, and completeness. The TDE will monitor the progress of the documentation and notify responsible parties when errors or incompleteness pose the threat of delaying a shipment. Finally, the TDE will also forward notification to the customs broker to request verification of the trade origination documents. The customs broker accesses the TDE via the same portal to review and verify the trade documentation. The TDE will also allow for collaboration between participating shippers, third-party logistics providers, carriers and customs brokers to define and document business requirements and risk assessment requirements.

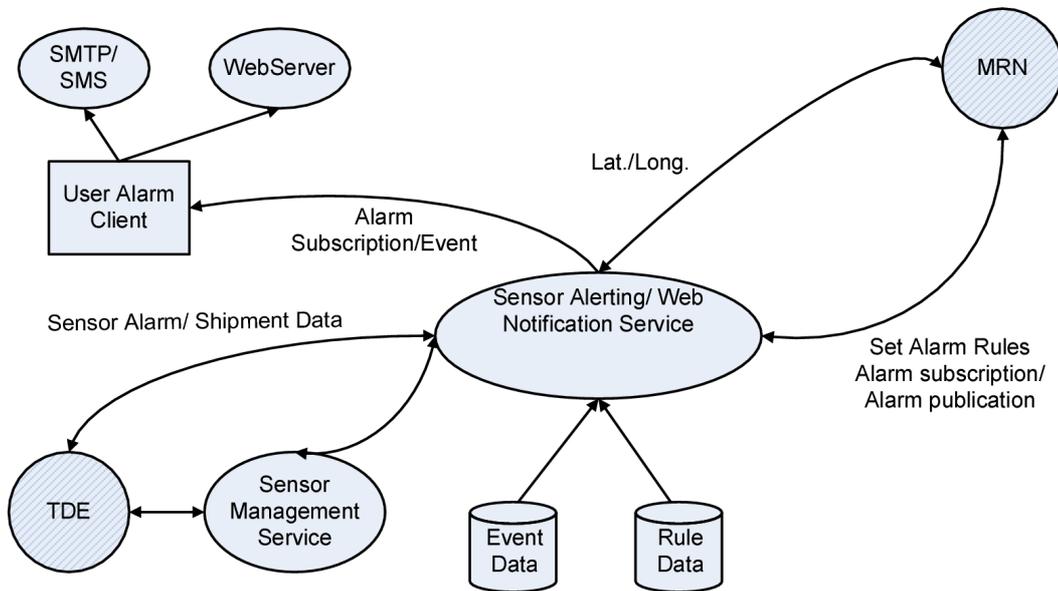


Fig. 2. Virtual Network Operations Center Architecture

B. Virtual Network Operations Center

The Virtual Network Operations Center (VNO) is the shipper's interface to clients and services that are outside the shipper's network—the TDE. The VNO is also the central decision and connection point for all of a shipper's MRNs. The VNO performs the following functions:

- Receives messages from the MRN.
- Obtains event-associated cargo information from the Trade Data Exchange (TDE).
- Makes decisions (using rules) on which MRN alarms are ignored or forwarded to decision makers, for example, a low battery alarm is sent to technical staff while an open/close event is sent to decision makers. These decisions are made using a complex event processor, Esper¹ [15], which takes into account shipping information as well as data (for example, geographical location) from current and past MRN alarms.
- Combines cargo information with an MRN alarm to form a VNO alarm message that is sent (by SMS and/or e-mail) to decision makers.

¹Esper was chosen because of the flexibility that it offers in defining rules. Furthermore, Esper was designed to operate on a stream of events, such as the set of incoming alarms from the MRN, and it has a rich syntax for specifying the relationship between elements of the input stream.

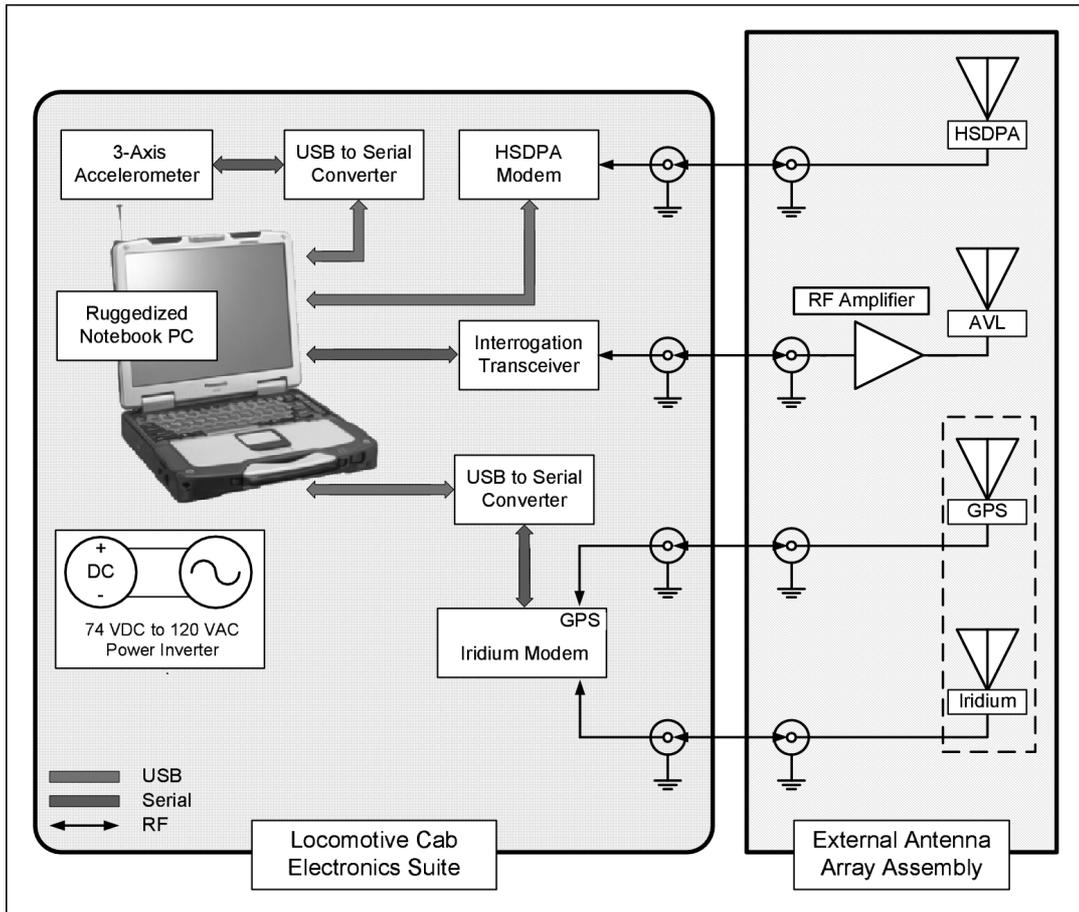


Fig. 3. TSSN Collector Node Hardware Configuration

- Forwards startMonitoring, stopMonitoring and getLocation instructions from a TDE client to the TSSN collector node.

Fig. 2 summarizes the VNOC and its components.

C. Mobile Rail Network

1) *Mobile Rail Network Hardware*: The MRN subsystem hardware consists of a set of wireless shipping container security seals and a TSSN collector node. The collector node is composed of two major sections: an electronics suite mounted in the locomotive cab and a remote antenna assembly that is magnetically attached to the exterior of the locomotive. Fig. 3 summarizes the key components of the TSSN collector node.

The electronics suite contains a power inverter, a security seal interrogation transceiver, a computing

platform, wireless data modems, a three-axis accelerometer, and a GPS receiver. The antenna assembly consists of three communications antennas, a GPS receiver antenna, and a bidirectional RF amplifier. A bundle of four 5.5 m (\approx 18 ft.) lengths of low insertion loss RF coaxial cable connect electronics suite devices to corresponding antennas.

Powering the TSSN collector node using the available 74 V dc locomotive power posed a challenge. The devices that comprise the node require four different dc input voltage levels, which ideally would be provided through the use of typical dc-to-dc conversion techniques, but in the interest of quickly deploying a proof of concept system, a 74 V dc to 120 V ac conversion was selected. Inverting the available dc power to 120 V ac allows plug-and-play use of the ac power converters provided with individual devices. A modified sine wave power inverter mounted in the electronics suite enclosure supplies 250 W of ac power capacity to the collector node.

The TSSN is designed to monitor and report security seal events including seal opened, seal closed, tampered seal, seal armed, and low battery warnings. Processing and storage of these events is tasked to a ruggedized notebook computer, which also serves as a portal to wireless communications resources. The three-axis accelerometer mounted in the electronics suite is monitored by the notebook computer, which logs movement data.

Container physical security is monitored using a system that was originally designed for tanker truck security [11]. The interrogation transceiver communicates with active and battery-powered wireless data seals over a wireless network using a 916.5 MHz signal. The interrogation transceiver communicates with the notebook computer via a serial data connection. The container seals use a secondary 125 kHz channel for communications with handheld programming equipment. The container seals are equipped with flexible wire lanyards that are threaded through container keeper bar lock hasps. Fig. 4 shows a container seal with a flexible wire lanyard.

Initial tests of the security seal and reader system revealed read ranges that were not adequate for the needs of this project. A bidirectional RF amplifier added between the interrogation transceiver and the antenna dramatically improved system performance, resulting in typical seal read ranges of several freight car lengths during field tests. It is understood that even with this improvement in read ranges we will not be able to monitor an entire train with our current technology choice. Different seal and/or mesh networking technologies would be needed for monitoring the entire length of typical cargo trains.

Communication between the MRN and the VNOC is accomplished using a HSDPA cellular data modem. An Iridium satellite modem is also available and is intended for use in remote locations that lack cellular network coverage. System communications using the Iridium modem are in the process of



Fig. 4. Container Seal

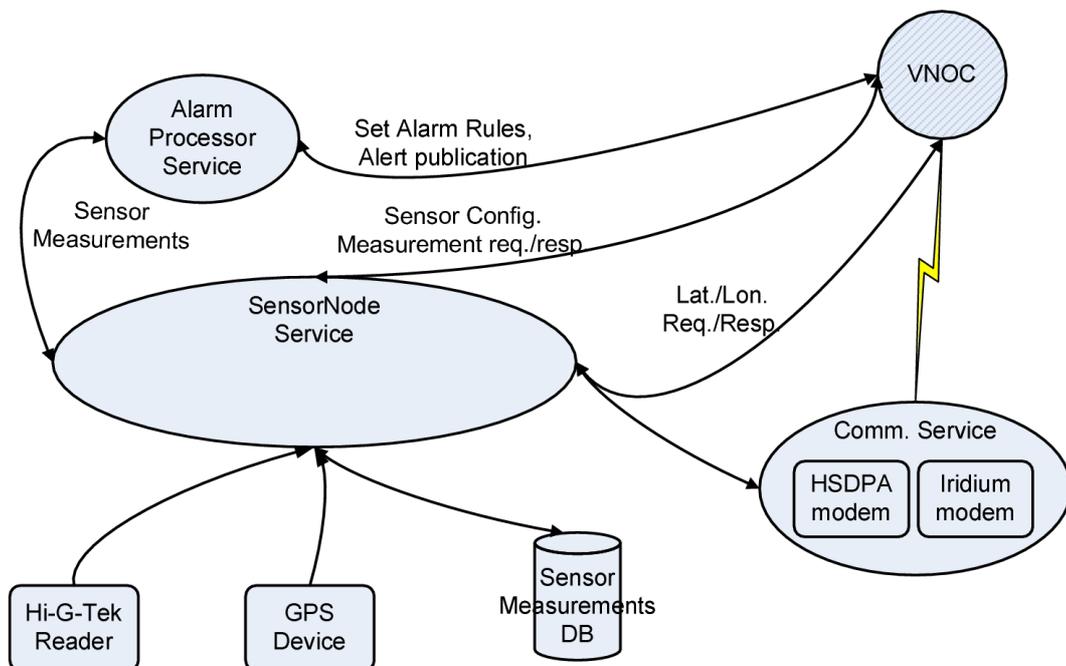


Fig. 5. Mobile Rail Network Collector Node Architecture

being implemented. The Iridium modem is a combination unit that includes a GPS receiver, which is used to provide the MRN position information.

2) *Mobile Rail Network Software*: The MRN software consists of a SensorNode service, an Alarm-Processor service, and a Communications service. The SensorNode service finds and monitors sensors which have been assigned to its control. The SensorNode service manages several sensor software plug-

ins, for example, a seal interrogation transceiver plug-in and a GPS device plug-in, that do all the work on behalf of the SensorNode service. During typical operation each container seal listens for interrogation command signals at three second intervals. The interrogation transceiver also queries the seals periodically (This took place every two minutes in these experiments.). In the event of a seal being opened/closed or tampered with, the seal immediately transmits a message to the SensorNode service running on the Collector Node. The message contains the seal event, a unique seal ID, and event time. The SensorNode service passes the seal message as an alert message to the service that has subscribed for this information.

The AlarmProcessor service determines messages from the SensorNode service that require transmission to the VNOC. Alarm messages include the seal event, event time, seal ID, and train's GPS location.

The Communications service currently logs the HSDPA signal strength. In the future we plan to build some intelligence into the Communication service so that it can switch between an Iridium and an HSDPA signal. Fig. 5 shows the key software functions of the MRN.

III. EXPERIMENTS

We have conducted two experiments to assess the suitability of the TSSN system for cargo monitoring as well as to collect data that would be used to guide the design of future cargo monitoring systems. In this section we present the experimental objectives and set-up, data collected during the tests, and issues that were encountered during the tests.

A. Road Test with Trucks

The first experiment was conducted on the roads around Lawrence, Kansas to determine the following:

- Approximate communication distances between the Hi-G-Tek sensors and the readers.
- Processing time through the system, including MRN, VNOC, and TDE, to SMS/e-mail messages to decision makers.
- Correct information is reported by the TSSN collector node including valid GPS coordinates.

The test was carried out using two pickup trucks, one of which had the locomotive cab electronics suite in the truck bed (The external antenna assembly was mounted to the tailgate of this truck.), while the other had a laptop that was used to control and monitor the VNOC. The VNOC was located in Lawrence, Kansas while the TDE was located in Overland Park, Kansas. Both trucks also had seals in their truck cabins so that seal open and close events could be simulated and reported. The seals were opened and closed at selected intersections along the test route that were easily identifiable on Google Maps [16].

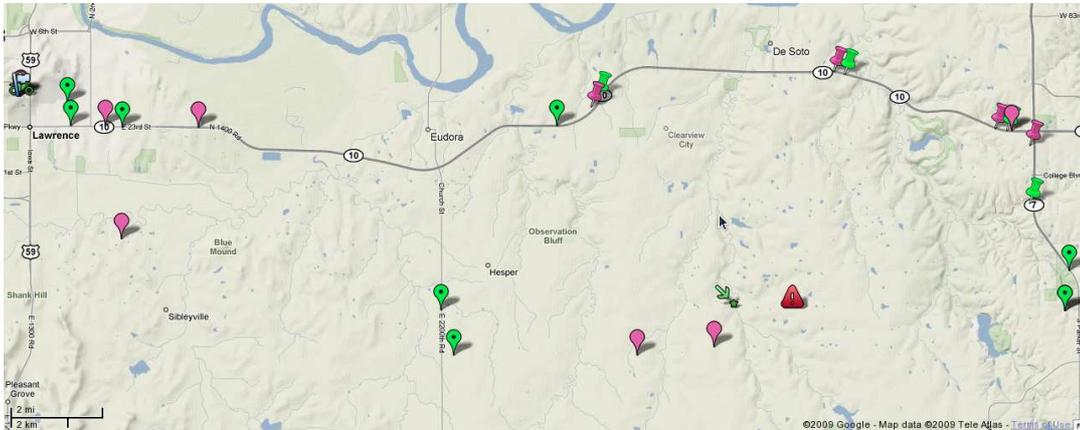


Fig. 6. Partial Map of Road Test with Event Annotations

Fig. 6 shows a trace of our route and the events overlaid on Google Maps. The pink tear drops indicate an open event, green tear drops a close event, pink tacks indicate a GPS lost signal, green tacks indicate where the GPS signal was regained, a red exclamation sign indicates where HSDPA connectivity was lost, and a green arrow indicates where HSDPA connectivity was regained. In summary, the road tests went well because open and close events were propagated correctly through our system. Furthermore, the system was able to recover from a dropped HSDPA connection.

Our test results indicate that all open and close events were reported as expected. The sensors and readers performed reliably. However, it is worth noting that the reader failed to read the sensors when the trucks were over 400 m apart on a hilly road. Finally, in our experiment we were able to combine sensor and shipment information to present reports to distributed decision makers. As a result, we conclude that the TSSN prototype worked in a mobile scenario.

During this experiment, system time on the TSSN Collector Node was maintained using the default mechanism in the Linux kernel (Even though we had a GPS receiver in the MRN, it was not used to maintain system time.). Analysis of event logs generated on the MRN and VNOC revealed that there was a significant amount of clock drift on the TSSN Collector Node during this relatively short (about 2.5 hours) trial. The time recorded at the VNOC for receipt of a message, in some cases, was earlier than the time recorded at the TSSN Collector Node for sending the message. Since time at the VNOC is controlled by a Network Time Protocol (NTP) [17] server, we conclude that the clock drift is occurring on the TSSN Collector Node. Correcting, or at least minimizing, the clock drift at the TSSN Collector Node is critical for evaluating overall TSSN performance, since the Collector Node is responsible for

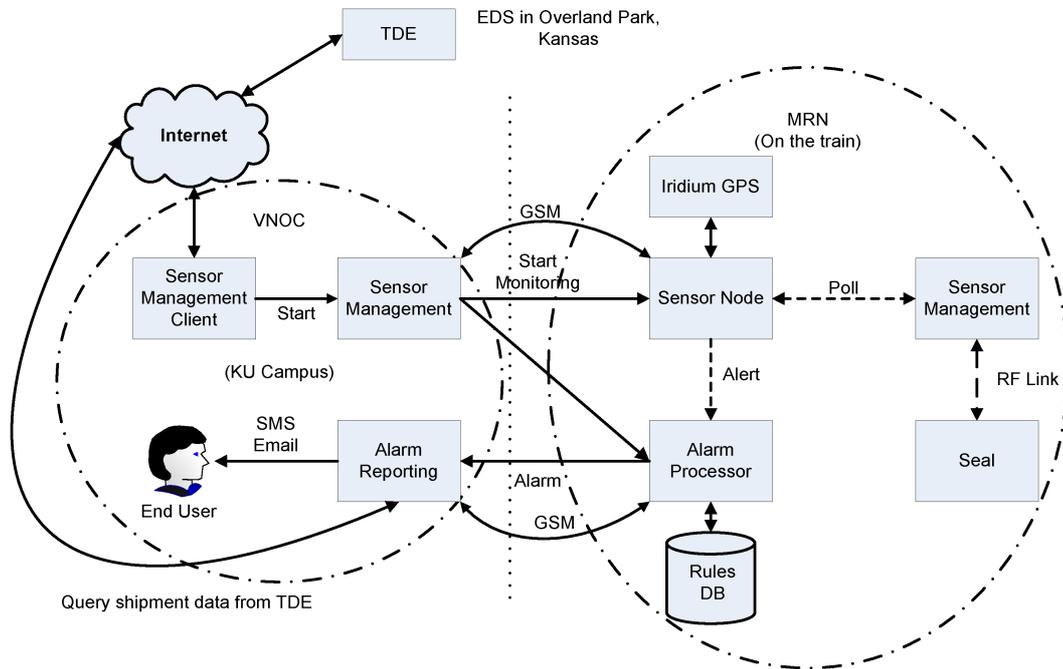


Fig. 7. Short-haul Rail Trial Configuration

establishing the time at which seal events occur. In the next version of the TSSN we have resolved the clock drift problem through a combination of software and hardware. It should be noted that in spite of the clock drift in the TSSN collector node we were able to correct for certain delays in our data. We discuss these corrections in Section V.

B. Short-haul Rail Trial

Our next experiment was carried on a train making an approximately 35 km (22 miles) trip from an intermodal facility to a rail yard. Our objectives in this experiment were the following:

- To determine the performance of the TSSN system when detecting events on intermodal containers in a rail environment.
- To investigate if decision makers could be informed of events in a timely manner using SMS messages and e-mails.
- To collect data that will be used in a model to investigate system trade-offs and the design of communications systems and networks for monitoring rail-borne cargo.

Fig. 7 shows the configuration used in the short-haul rail trial. In this experiment the VNOG was located in Lawrence, Kansas, the TDE was located in Overland Park, Kansas, while the TSSN collector

```

NOC_AlarmReportingService:
Date-Time: 2009.01.07 07:12:17 CST/
          2009.01.07 13:12:17 UTC
Lat/Lon: 38.83858/-94.56186, Quality: Good
http://maps.google.com/maps?q=38.83858,-94.56186
TrainId=ShrtHaul
Severity: Security
Type: SensorLimitReached
Message: SensorType=Seal
        SensorID=IAHA01054190
        Event=Open Msg=
NOC Host: laredo.ittc.ku.edu

Shipment Data:
Car Pos: 3
Equipment Id: EDS 10970
BIC Code: ITTC054190
STCC: 2643137

```

Fig. 8. Partial Screen Shot of e-mail Message Sent During Trial

node was placed in a locomotive and used to monitor five seals placed on intermodal shipping containers and in the locomotive.

During the experiment, events were simulated by breaking and closing a seal (sensor) that was kept in the locomotive. The VNOC reported these events to decision makers using e-mail and SMS messages. Fig. 8 shows the content of one of the e-mail messages that was sent to the decision makers.

In Fig. 8, the sensor ID, latitude and longitude data, and event type come from the MRN, while the shipment data comes from the TDE. The VNOC combines these pieces of information into an e-mail message that also includes a link to Google Maps, so that the exact location of the incident can be visualized. The ultimate value of the TSSN is getting this type of message to the decision maker.

During the test the reader lost communication with the seals for a brief period along the route. Future experiments will determine whether or not this loss of connectivity was due to RF interference. In spite of this, the experiment was a success as events were detected by the seals and reported to decision makers using both e-mail and SMS messages. Extensive log files were collected during the test and they are being postprocessed to obtain data on TSSN system performance.

IV. POSTPROCESSING OF EXPERIMENTAL DATA

In this section we discuss the framework for postprocessing the results of our experiments. Following the short-haul rail trial we collected log files from the VNOC, MRN, and TDE. These log files contain data on message sizes, timestamps, event type, message type (incoming/outgoing) amongst other data elements. Our objective was to postprocess these files to evaluate the performance of the TSSN system.

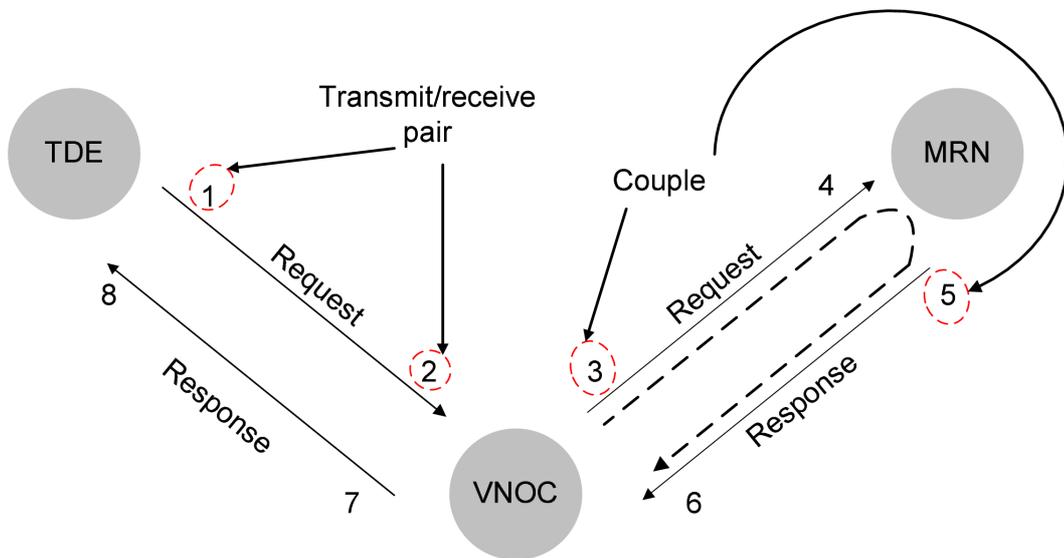


Fig. 9. LogParser Framework Showing Message Couples and Transmit/receive Pairs

Postprocessing of log files from geographically distributed computers was accomplished using a Java library (LogParser) that was developed in-house. First, the library read in all available information in each log file including time, message size, from and to addresses, as well as the original SOAP message. Information from all (MRN, VNO, and TDE) of the log files in an experiment was combined into a single collection of log entries. We expect that every message transmitted in the TSSN should result in at least two log entries—a transmit log entry (at the originating entity) and a received log entry (at the receiving entity). The LogParser library identified log entries as:

- Transmit/receive pairs, that is, the outgoing and incoming log entries with the same SOAP WS-Addressing (The SOAP WS-Addressing specification “provides transport-neutral mechanisms to address Web services and messages [18].”), and
- Couples, that is, SOAP request/response message pairs.

Fig. 9 shows the relationship between log entry couples and transmit/receive pairs. Suppose the TDE sends a message to the VNO requesting the current MRN location. The circled “1” and “2” in Fig. 9 denote the log entries representing message transmission from the TDE and receipt of this same message at the VNO. Couples are a bit more involved; much of the communication between client/server is based on a request/response model. As a result, there are two related messages which contain additional information to establish their relationship:

- 1) REQUEST: from client to server asking for something; and
- 2) RESPONSE: from server back to the client with the response.

Log entry couples are marked by the records for the outgoing request and response messages. Consequently, the circled “3” and “5” in Fig. 9 constitute the log entry couple for the VNOC forwarding the location request message to the MRN and the MRN’s origination of a response respectively. Using the receive pairs for records “3” and “5”, we can also identify entries “4” and “6.”

With this framework, programs were written against the log entry collection to extract the number of messages sent by each service, request/response time for messages, processing time at either the MRN, VNOC, or TDE, the time that messages were carried by the network, and message sizes. Additional information, for example, latitude, longitude, sensor IDs, and event timestamps, could be extracted from the SOAP message using XPath expressions. XML Path language (XPath) allows for addressing “parts of an XML document [19].” XPath also provides “basic facilities for manipulation of strings, numbers and booleans [19].”

V. RESULTS

In this section we discuss the results of the TSSN system evaluation based on the short-haul rail trial. One objective of our experiments was to determine whether decision makers could be notified of events in a timely manner. Due to significant clock drift in the TSSN collector node, we can only present an estimate of the time taken for an event report to travel from the MRN to the VNOC. However, exact time values can be computed for other TSSN component interactions.

In addition, we present time statistics on interactions between the TSSN component subsystems. These statistics hint at how the aggregate time from event detection to decision maker notification is distributed among the various services and communication links in the TSSN. With this information we will be able to guide system refinements to further reduce the overall time. In our analysis we present results on the following:

- **Service request processing time.** This is the time between when a service receives a request and when a response message is composed. Using Fig. 9, this time can be computed as the time difference between log entries “5” and “4.”
- **Request/response time.** This is the time taken to get a response from a remote service, including the processing time. Using Fig. 9, this time can be computed as the time difference between log entries “6” and “3.”

- **Network time.** This is the time taken to get a response from a remote service, excluding the processing time. This can be computed by subtracting the service request processing time from the request/response time.

Our time analysis in this section will examine request/response messages going from the VNOC to the MRN back to the VNOC, from the TDE to the VNOC back to the TDE, and from the VNOC to the TDE back to the VNOC.

A second objective for the short-haul rail trial was to confirm that messages were being passed correctly between the different components of the TSSN. As a result, we provide a summary of the messages exchanged between different parts of the TSSN system.

The last objective of the short-haul rail trial was to collect data that will be used in a model [20] to design systems for monitoring rail-borne cargo and determine trade-offs. Message sizes and interevent times are two components of this model. As a result, we present a table summarizing the message size² statistics between different components of the TSSN. We also present histograms summarizing message intercommand and interalarm times at the MRN. Both of these times are needed, in conjunction with message sizes, to compute the cost of reporting messages (Both the alarms and commands were simulated in our experiment; deployed systems will show different statistics for intercommand and interalarm times.).

Finally, this section also presents results showing how HSDPA signal strength varied with time during the short-haul test. The HSDPA signal strength results may be used to help determine when to switch between HSDPA and Iridium.

A. VNOC to MRN to VNOC Interaction

The statistics on VNOC to MRN to VNOC interaction allow us to draw conclusions on the time taken to complete one component of processing `startMonitoring`, `stopMonitoring`, and `getLocation` messages. In addition, these statistics allow us to gain insight into the one-way network delay from the TSSN collector node to the VNOC—a delay that is one component of sending an event report from the MRN to the VNOC. Fig. 10a is a histogram showing the request/response time for messages going from the VNOC to the MRN and back to the VNOC. Using Figs. 10b and 11 we cannot conclude that the request/response time is dominated by the processing time. In this instance the request/response time appears almost equally split between the processing and network times. Note that in Fig. 11 our minimum is 0 within the resolution of the experiment.

²It should be noted that message sizes can be computed *a priori*; however, the distribution of these messages cannot be determined beforehand.

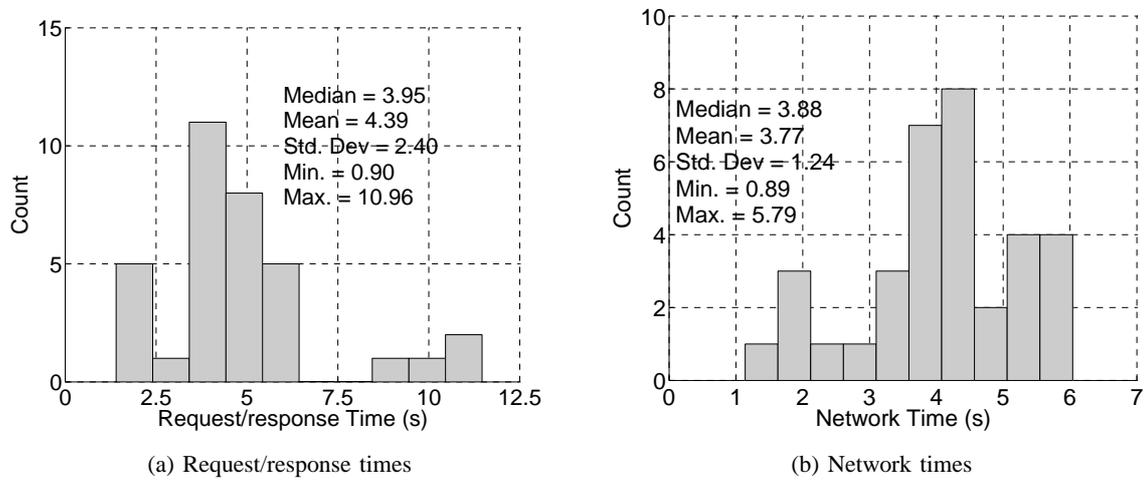


Fig. 10. Request/response and Network Times from VNOC \rightarrow MRN \rightarrow VNOC

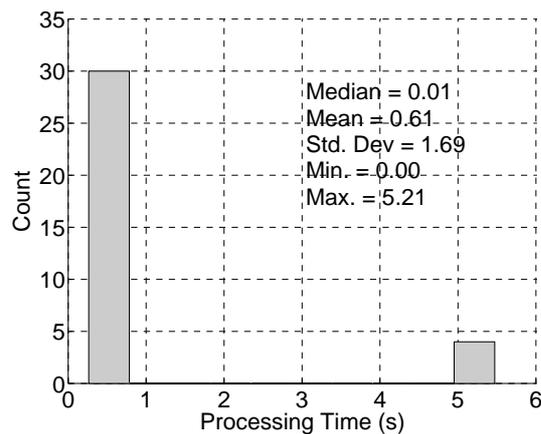


Fig. 11. Processing Times at MRN

Due to clock drift in the TSSN collector node, we are unable to obtain statistics on the one-way network delay for sending an MRN_Alarm message—which indicates an event at a sensor—to the VNOC. However, it is reasonable to assume that the MRN \leftrightarrow VNOC links are symmetric thus, the one-way delay from the MRN to the VNOC is approximately 1.89 s.

B. Elapsed Time from Alert Generation to AlarmReporting Service

The time taken for the TSSN to process an event report is an important metric in evaluating this system. Furthermore, demonstrating that this metric is of the order of several seconds can help convince decision makers of the TSSN's utility. Due to clock drift in the MRN we cannot compute an exact value

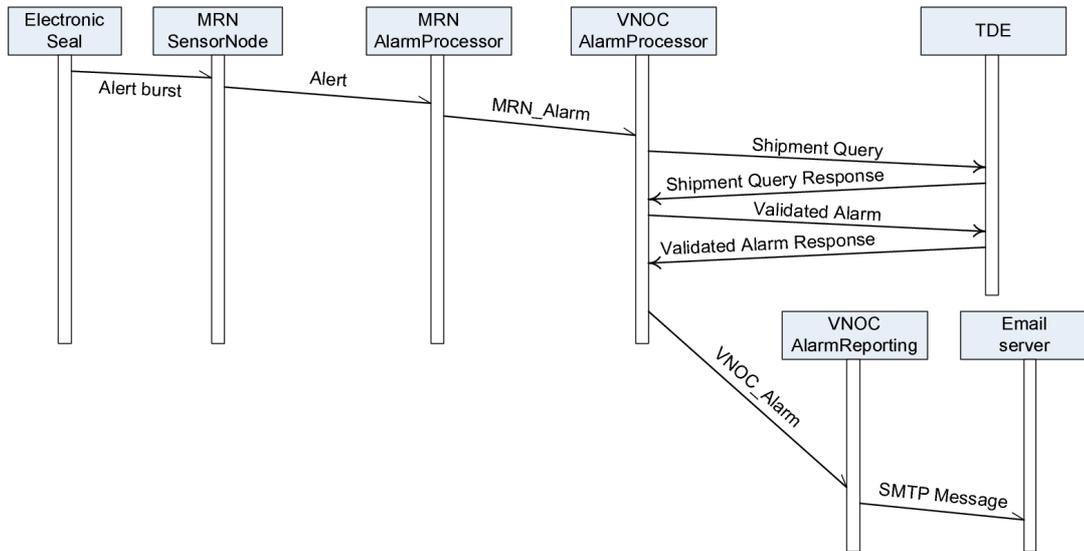


Fig. 12. Sequence Diagram with Messages Involved in Decision Maker Notification

for time taken for an MRN_Alarm to go from the MRN to the VNOC. However, we can use the 1.89 s estimate from the previous subsection as a reasonable value for this network delay. Fig. 12 shows the rest of the messages involved in notifying a decision maker of an event at a seal.

Given a system with no clock drift and an identifier that relates Alerts, MRN_Alarms, and NOC_Alarms, we can easily compute the time taken to notify decision makers by subtracting the log entry timestamp for the Alert message when it is generated at the SensorNode service from the log entry timestamp for the NOC_Alarm when it arrives at the VNOC AlarmReporting service. Unfortunately, we do not have a unique identifier and there is clock drift in the MRN. As a result, we generated the results in this subsection as follows: three sets were created comprising of all NOC_Alarms, all MRN_Alarms, and all Alerts respectively. For each NOC_Alarm, the set of MRN_Alarms was scanned for a message having the same seal ID and event timestamp without being a status message. The time difference between the log entries for the incoming message at the VNOC AlarmProcessor and the VNOC AlarmReporting services gives us the period taken for the VNOC AlarmProcessor to process any shipment queries, store alarms, and transmit the message to the VNOC AlarmReporting service. To this value we add our estimated one-way MRN_Alarm network delay of 1.89 s. Next, we search the set of Alerts for a message having the same seal ID and event timestamp without being a status message. The time difference between the log entries for the outgoing Alert message at the MRN SensorNode service and the outgoing MRN_Alarm at the MRN AlarmProcessor service gives us the elapsed time between the two services as well as

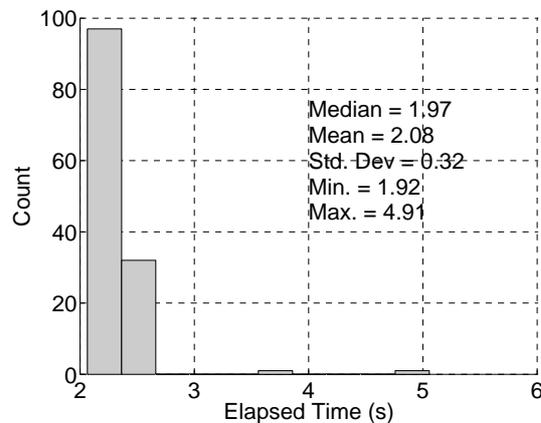


Fig. 13. Elapsed Time from Alert Generation to VNOc AlarmReporting Service

the processing delay at the MRN AlarmProcessor service. This period is added to the two previously calculated time periods.

Fig. 13 is a histogram showing the distribution of the elapsed time from when the MRN SensorNode generates an alert until the VNOc AlarmReporting service receives the notification. By performing this analysis we see that on average it takes about 2 s for messages to get from the MRN SensorNode service to the VNOc AlarmReporting service. Thus, we conclude that the time taken to process events in the TSSN is not an impediment to timely notification of decision makers.

C. End-to-end Time from Event Occurrence to Decision Maker Notification

An important metric for TSSN performance is the time between event occurrence until a decision maker is notified using an SMS message. Since this time is a random variable, we can create other metrics based on this time that return the probability that the TSSN can deliver notification within a specified interval. The components of the end-to-end time include:

- Time between between event occurrence and when the MRN SensorNode service generates the related event alert.
- Time from alert generation to the VNOc AlarmReporting service. Based on the previous subsection, this is about 2.08 s on average, while the longest time observed was 4.91 s.
- Time taken for the VNOc AlarmReporting service to process and send an e-mail message to an e-mail server.
- Time taken by the SMS vendor to get the message to a decision maker's phone.

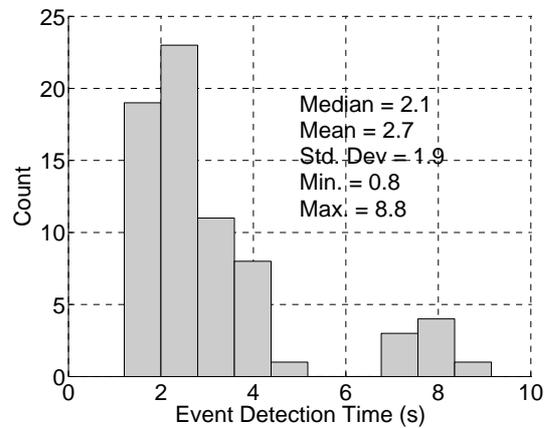


Fig. 14. Elapsed Time from Event Occurrence to Alert Generation

To overcome inaccurate clocks in the seals, we set up a lab experiment to determine the elapsed time between event occurrence and the TSSN's generation of the related event alert. In this experiment, a stopwatch was started when a seal was either broken or closed; when the MRN SensorNode service generated an event alert the stopwatch was stopped. Fig. 14 is a histogram showing the time distribution between event occurrence and the MRN SensorNode service generating an alert. From Fig. 14 we see that the longest observed time between event occurrence and the MRN generating an Alert is about 8.8 s. Furthermore, it takes about 2.7 s on average.

A second experiment was carried out to determine the elapsed time between the VNOC AlarmReporting service's transmission of a VNOC alarm message and the decision maker receiving event notification. In this experiment a client program was written to send messages to the VNOC alarm reporting service. A stopwatch was started when the VNOC sent an alarm to a decision maker and the stopwatch was stopped when the decision maker's phone received an SMS message. Table I summarizes the statistics for delivery of alarm messages for different carriers. Fig. 15 is a histogram showing the distribution of the time taken to deliver alarm messages to decision makers.

From Table I we see that even though SMS was not designed as a real-time system, it provides excellent notification for our purposes; since most of our messages were delivered within a short time.

TABLE I
SUMMARY OF TIME TAKEN TO DELIVER SMS MESSAGES

Carrier	Min./s	Max./s	Mean/s	Median/s	Std. Dev./s	n
Telco 1	5.9	18.4	12.2	11.8	2.9	30
Telco 2	5.2	30.4	8.8	7.8	4.5	30
Telco 3	7.1	43.0	10.8	9.0	6.7	30
Telco 4	5.9	58.7	15.7	11.1	11.1	30

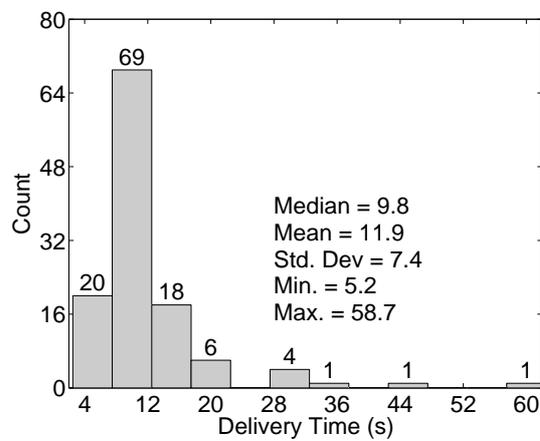


Fig. 15. Time Taken to Deliver SMS Messages for All Carriers

Combining all of these experimental results, we see that in the longest observed case it can take just over one minute³ to notify decision makers of events. Most of this time is spent delivering an SMS message to the decision maker, so we conclude that the TSSN provides a mechanism for timely notification of decision makers.

D. TDE to VNOC to TDE Interaction

The statistics on TDE to VNOC to TDE interactions allow us to draw conclusions on the time taken to initiate and process startMonitoring, stopMonitoring, getLocation, and setAlarmSecure messages. These messages are all forwarded to the MRN, and the VNOC returns the response that it receives from the MRN. To the TDE, all the elapsed time from when the VNOC receives a message from the TDE until

³This time is broken out as follows: in the longest observed times in our experiments it took approximately 8.8 s between event occurrence and the TSSN generating an alert; 2) it took approximately 4.91 s for an alert message to go through the TSSN until notification was sent to decision makers; and 3) it took up to 58.7 s to deliver an SMS message to decision makers.

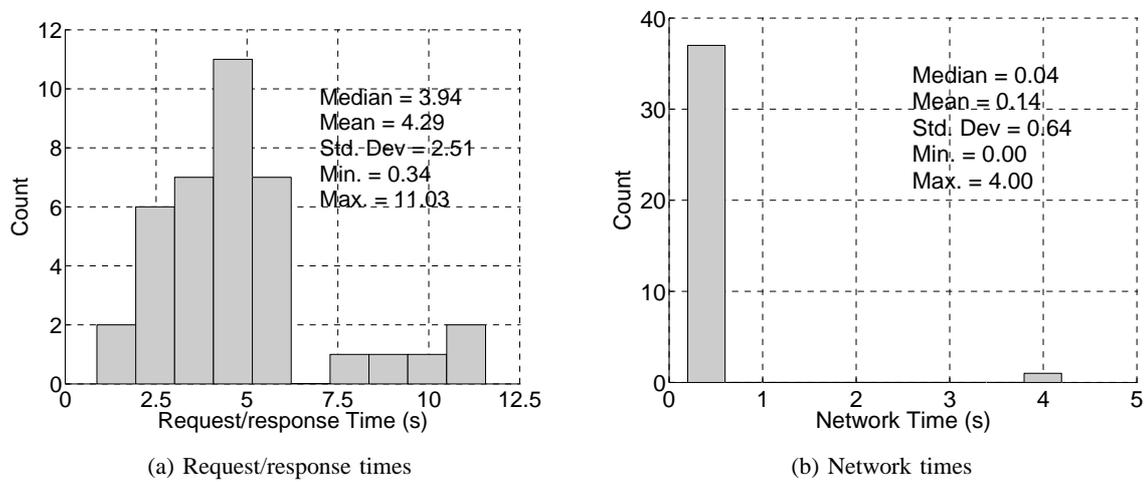


Fig. 16. Request/response and Network Times from TDE \rightarrow VNOC \rightarrow TDE

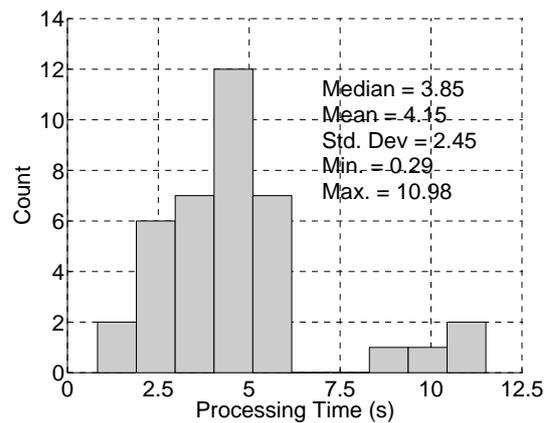


Fig. 17. Processing Times at VNOC

the VNOC sends a response is processing time at the VNOC, even though part of that time is spent forwarding a response to the MRN and waiting for a response. Fig. 16a is a histogram showing the request/response time distribution for messages going from the TDE to the VNOC and back to the TDE. Using Figs. 16b and 17 we conclude that the request/response time is dominated by the processing time at the VNOC. This conclusion is supported by the request/response time result from Section V-A, which showed times of up to 10.96 s.

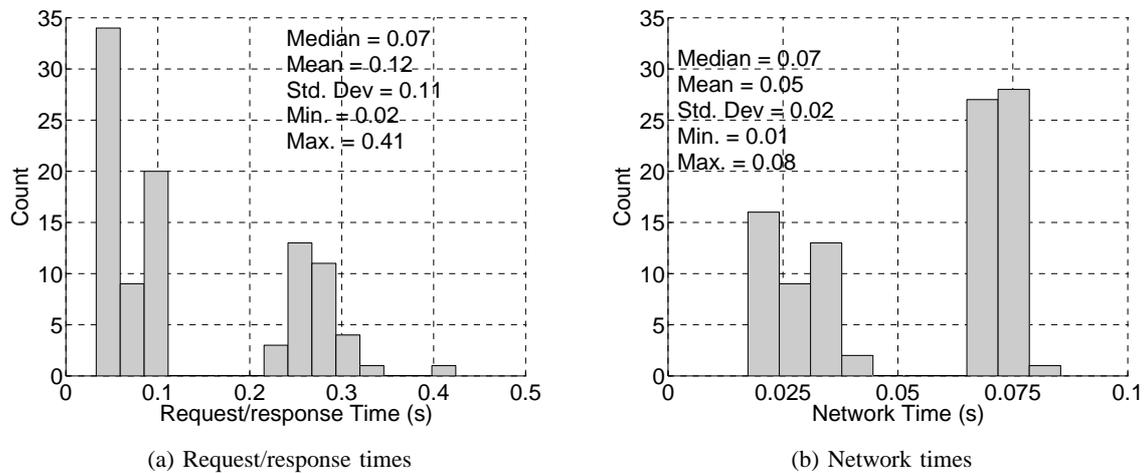


Fig. 18. Request/response and Network Times from VNOC \rightarrow TDE \rightarrow VNOC

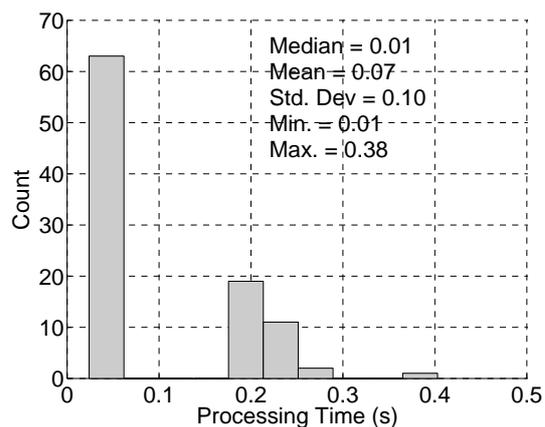


Fig. 19. Processing Times at TDE

E. VNOC to TDE to VNOC Interaction

The statistics on VNOC to TDE to VNOC interactions allow us to draw conclusions on the time taken for the TDE to store alarm messages and execute shipment queries. Both of these actions are carried out when the VNOC alarm processor service is about to send an alarm to the VNOC alarm reporting service. Fig. 18a is a histogram showing the request/response times for messages going from the VNOC to the TDE and back to the VNOC. From Fig. 18a we conclude that on average it takes approximately 0.12 s to either store an alarm message or get a shipment query response. Using Figs. 18b and 19 we find that the request/response time is dominated by the processing time, just as we found in Section V-D.

TABLE II
SUMMARY OF TIME STATISTICS

Description	Min./s	Max./s	Mean/s	Median/s	Std. Dev./s
Request/response times from VNOC → MRN → VNOC	0.90	10.96	4.39	3.95	2.40
Network times from VNOC → MRN → VNOC	0.89	5.79	3.77	3.88	1.24
Processing times from VNOC → MRN → VNOC	0.00	5.21	0.61	0.01	1.69
Event occurrence to alert generation	0.81	8.75	2.70	2.13	1.86
Alert generation to VNOC AlarmReporting Service	1.92	4.91	2.08	1.97	0.32
Request/response times from TDE → VNOC → TDE	0.34	11.03	4.29	3.94	2.51
Network times from TDE → VNOC → TDE	0.00	4.00	0.14	0.04	0.64
Processing times from TDE → VNOC → TDE	0.29	10.98	4.15	3.85	2.45
Request/response times from VNOC → TDE → VNOC	0.02	0.41	0.12	0.07	0.11
Network times from VNOC → TDE → VNOC	0.01	0.08	0.05	0.07	0.02
Processing times from VNOC → TDE → VNOC	0.01	0.38	0.07	0.01	0.10

F. Summary of Time Statistics

Table II summarizes the statistics shown in each of the time histograms in this section. Note that there are no results for the MRN to VNOC to MRN interaction. This is due to two reasons: first, clock drift in the MRN prevents us from computing a one-way network delay. Secondly, the MRN only generates response messages. There are no request messages originating from the MRN that could be used in a log entry couple to calculate request/response or processing times.

G. Messages by Schema Element

One objective of our postprocessing was to determine if messages were being passed correctly between the TSSN components. Fig. 20 shows the messages exchanged by various components of the TSSN system. From Table III we see that all messages are logged correctly in the log files. For example, the VNOC sent 63 shipment query requests (TDEService/ShipmentQuery) to the TDE and received 63 shipment query responses (TDEService/ShipmentQueryResponse). Similarly, the VNOC sent 33 validated alarms to the TDE and got 33 validated alarm responses from the TDE. From Table III we also see that some of the messages are being filtered by the system. For example, the MRN SensorNode service reports 546 alerts to the MRN Alarm Processor. Only 131 alerts met the MRN subsystem's rules and these were forwarded to the VNOC's Alarm Processor. All of the alarms received by the VNOC alarm processor met the necessary rules so that they could be forwarded to decision makers as SMS or e-mail messages.

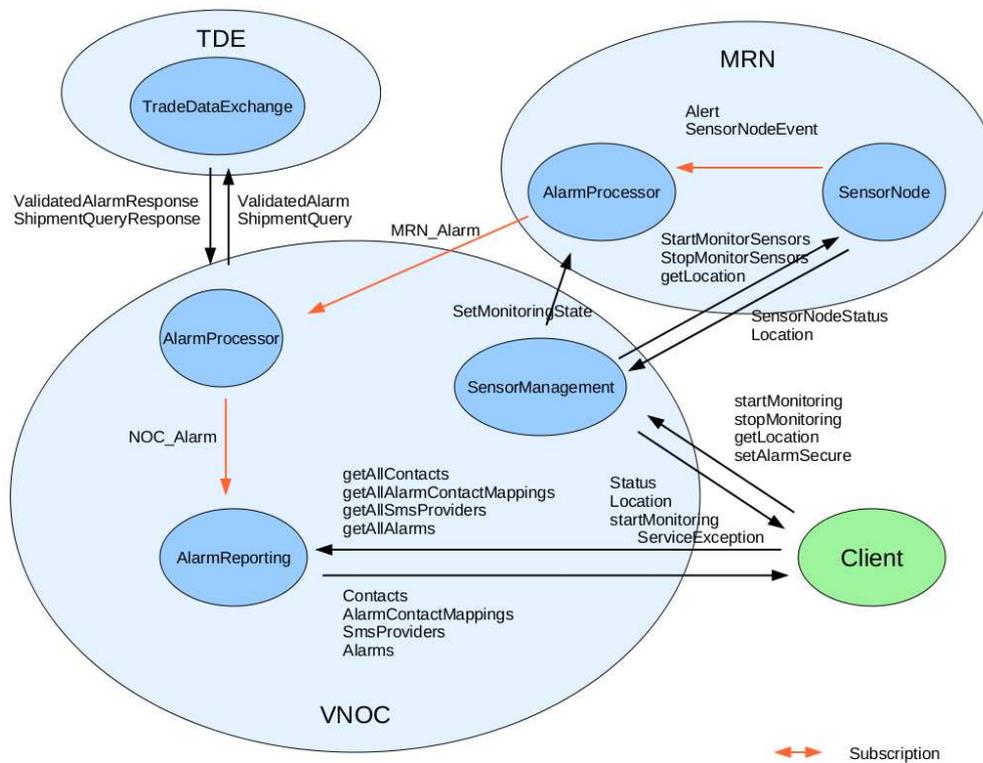


Fig. 20. Component Interactions in the TSSN

H. Message Sizes

A model [20] is under development to determine system trade-offs as well as optimal or near-optimal sensor locations when using a rail-borne cargo monitoring system. The cost of transmitting a message from the train to an operations center is one component of this model. This transmission cost, in turn, depends on the average message length transmitted from the train and the frequency at which these messages are generated. This section presents results on message sizes between the MRN and the VNOG, while Section V-I presents results on intercommand and interalarm times for messages exchanged between the MRN and the VNOG.

Table IV summarizes the message size statistics for all the messages exchanged in the TSSN. Additional analysis (which is omitted here) showed that the message size groupings typically coincided with the number of message types exchanged on each link. For example, the MRN sent three different message types to the VNOG, and review of message size data between the MRN and VNOG confirmed three distinct message types.

TABLE III
NUMBER OF MESSAGES GENERATED BY SCHEMA ELEMENT

Schema Element	Nbr of Messages
Subscribe	1
SubscribeResponse	1
ns:startMonitoring	1
ns:stopMonitoring	2
ns:setAlarmSecure	4
tssn:Status	8
ns:getLocation	30
tns:Location	30
tns:SetMode	1
mrnsnx:StartMonitorSensors	2
mrnsnx:StopMonitorSensors	2
mrnsnx:SensorNodeStatus	4
urn:startMonitoringServiceException	1
mrnsnx:getLocation	30
mrnsnx:Location	30
ns:SetMonitoringState	4
sas:Alert	546
mrnpub:MRN_Alarm	131
TDEService/ValidatedAlarm	33
TDEService/ValidatedAlarmResponse	33
TDEService/ShipmentQuery	63
TDEService/ShipmentQueryResponse	63
nocpub:NOC_Alarm	131

TABLE IV
SUMMARY OF MESSAGE SIZE STATISTICS

Description	Min./bytes	Max./bytes	Mean/bytes	Median/bytes	Std. Dev./bytes
TDE → VNOC	846	1278	874.7	848	96.8
VNOC → TDE	968	975	971.5	971	2.6
VNOC → MRN	650	1036	690.8	650	101.5
MRN → VNOC	799	1560	1419.2	1536	237.1

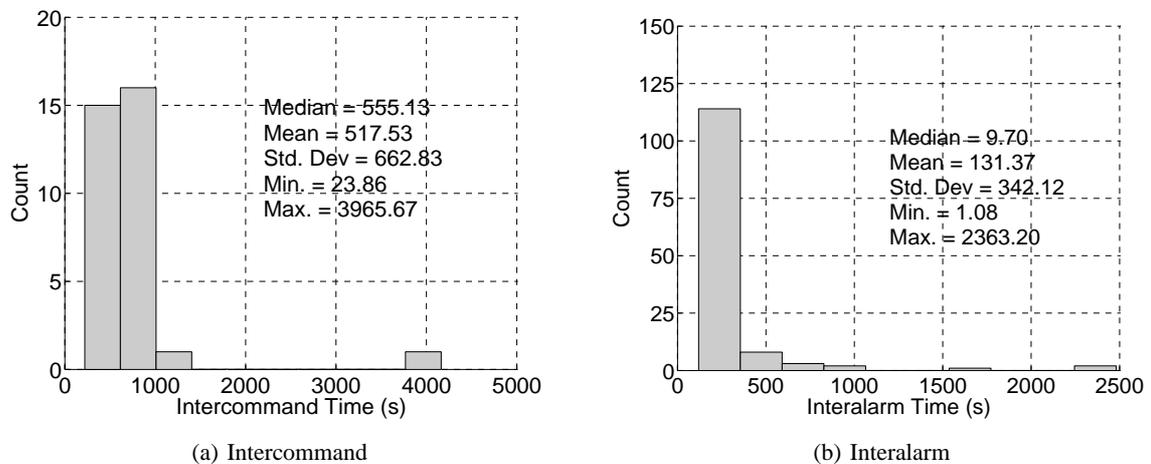


Fig. 21. Intercommand and Interalarm Times at MRN

I. Intercommand and Interalarm Times

The data collected from these experiments will be used in a model to determine system trade-offs when using a rail-borne cargo monitoring system. Communication costs in this model depend on the frequency (interalarm time) with which messages need to be reported, the mode of communications, as well as the message length in bytes. The intercommand time is included in this analysis because incoming messages may also be billed. Figs. 21a and 21b summarize the inter-command and inter-alarm times respectively at the MRN. The data presented here can be used as a starting point for adaptive MRN Communications service algorithms that “call” the VNOC periodically.

J. HSDPA Signal Strength

In later iterations of the TSSN we plan to switch between HSDPA and Iridium signals. HSDPA signal strength traces can help us tune algorithms that determine when to make the signal switch. Work still needs to be done to develop these algorithms. In this subsection we show how HSDPA signal strength varied with time during the short-haul rail trial.

During the short-haul rail trial, HSDPA was used to transmit messages from the MRN to the VNOC. As a result, the HSDPA signal strength was also recorded in the MRN log file. The LogParser library was used to extract this information, and HSDPA signal strength was plotted against the number of seconds from the start of the experiment in Fig. 22. The signal strength trace shown in Fig. 22 reflects our observations from the trial. During the first 80 minutes of the experiment the HSDPA signal trace remains fairly constant, since the train is stationary. Once the train begins to move the HSDPA signal

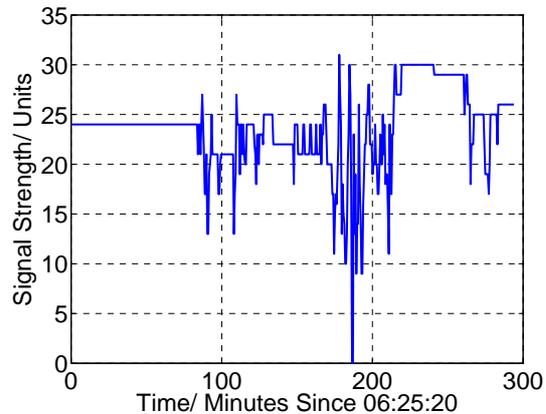


Fig. 22. HSDPA Signal Strength versus Time

strength varies with time. We notice two other flat portions on the trace at about 220 and 240 minutes. As before, the train was stationary at these points. Fig. 23 shows how the HSDPA signal strength varied with location over the duration of our experiment. The placemarks (colored tear drops) in Fig. 23 represent the HSDPA signal strength, which is given in a 0–30 scale with 0 representing no signal and 30 showing maximum signal strength. A red placemark denotes a signal strength of less than 10, a yellow placemark denotes a signal strength between 10 and 14, a blue placemark denotes signal strength between 15 and 19, a green placemark denotes a signal strength between 20 and 24, and a purple placemark denotes a signal strength of over 25.

VI. IMPACT ON SYSTEM MODELING

New models are needed to characterize rail-based cargo monitoring systems. These models can be applied, along with optimization theory, to determine system trade-offs when monitoring cargo in motion. The models can also be used to find the best locations for sensors in a rail-based sensor network as well as to guide the design of future cargo monitoring systems. In Section V we presented experimental results from a short-haul rail trial of the TSSN. There is ongoing work [20] to determine optimal or near-optimal placement of sensors for monitoring rail-borne cargo. Our objective in this research is to develop extensible models that can give the best (cheapest) system design while preserving the shipper’s desired level of security. Given a set, C , of containers to be placed on a train, a set, L , of possible locations for the containers on the train, a set, S , of sensors, and a set, R , of network elements, we can create a mapping, M_C , using Lai *et al.*’s [21] approach, that maps containers to locations on a train. We can also create mappings, M_R , and M_S , that map network elements and sensors, respectively, to

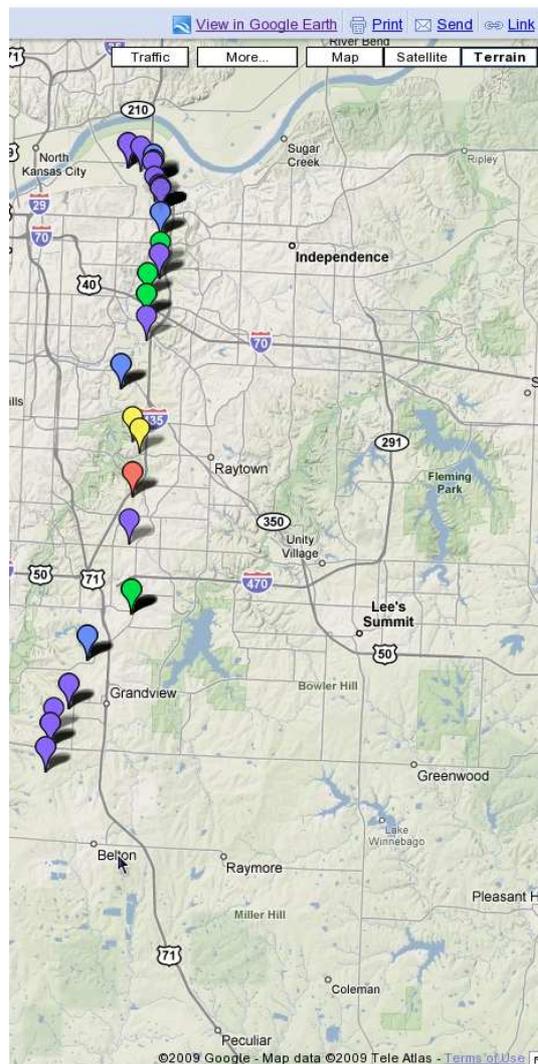


Fig. 23. HSDPA Signal Strength and Geographical Location

locations on the train; alternatively, M_S may map sensors to containers. Given these mappings we can create a function, f , that takes as input the sets of containers, locations, sensors, and network elements, as well as the mappings described above and returns a system cost metric. The goal of this research is to develop such a function, use the results from Section V in making the model more realistic, and determine if this function can be minimized in polynomial time.

To this end two models have been built to compute the cost metric of a cargo monitoring system. The models have the following general format: Given a list of parameter values p_1, p_2, \dots, p_n (such as the container values, savings resulting from detecting events at containers, request/response times

from VNOC \rightarrow MRN \rightarrow VNOC, and message sizes on the VNOC \leftrightarrow MRN link), we define variables x_1, x_2, \dots, x_n (such as a variable that indicates if a sensor is placed on a certain container). We also define a function $f_o(\bar{x}; \bar{p})$ that depends on the parameters and variables to return the system cost. (One of the components of f_o includes the cost of transmitting event reports from the MRN to the VNOC.) Our goal in this research is to minimize this objective function subject to the constraints⁴ specified by the system designer. These models will be used to determine system trade-offs, such as a rail-mounted or trackside deployment of network elements.

VII. REFINEMENTS BASED ON PRELIMINARY RESULTS

In preparation for additional rail trials, a GPS receiver change has been implemented and other MRN hardware system upgrades have been planned. To avoid conflicts between GPS receiver operation and Iridium modem use, a high performance GPS receiver has been installed on the External Antenna Assembly to replace the Iridium modem GPS functionality. The time drift issue mentioned in Section III-A will be resolved by using the high performance GPS receiver to get high quality local time. Pulse per second (PPS) output from the GPS receiver will be used as an input to the NTP server running on the TSSN collector node.

In addition to a new GPS receiver, proposed enhancements to the MRN hardware prototype include moving communications devices from the Electronics Suite to the External Antenna Assembly. The current hardware configuration suffers from the insertion loss of the long RF cable connections. Collector node interconnections between the locomotive cab and the external assembly would change from an RF signal connection to a DC power and data bus connection for each device. Moving the wireless modems and interrogation transceiver as close as possible to the corresponding antennas is expected to provide very significant performance improvements.

Postprocessing of the log files also indicated that a unique identifier—perhaps composed of a timestamp and counter—is needed in the Alert, MRN_Alarm, and NOC_Alarm messages to trace an Alert message through the TSSN. This identifier can also be used in the future to locate MRN_Alarm messages that need to be retransmitted to the VNOC following a loss of connectivity. Finally, the identifier can be used to mark previously processed messages so that the VNOC does not process the same message more than once.

⁴Some of these constraints specify valid placements for sensors and associated communications infrastructure. The constraints might also require that events at certain containers be detected with a certain probability and reported within a given time interval with specified probability.

Prior to deploying the TSSN system, further research is needed to address issues including:

- Communications infrastructure for whole train monitoring.
- Backhaul communications, including choosing when to switch between HSDPA and Iridium connections.
- Development and use of a model to seek trade-offs when monitoring rail-borne cargo.

The desired result of our research is a standards-based open environment for cargo monitoring with low entry barriers to enable broader access by stakeholders while showing a path to commercialization.

VIII. CONCLUSION

In this paper we have presented results from preliminary field trials of the TSSN (Transportation Security Sensor Network). Within the TSSN framework we have successfully combined sensor and shipment information to provide event notification to distributed decision makers. This paper has shown results documenting the interactions between the different components of the TSSN. Based on our experiments and evaluations we believe that the TSSN is viable for monitoring rail-borne cargo. These beliefs are based on the following: first, we have successfully demonstrated that alert messages can be sent from a moving train to geographically distributed decision makers using either SMS or e-mail. Second, based on the experiments reported here, we are able to detect events and notify decision makers in just over one minute. Thus, we conclude that the TSSN provides a mechanism for timely notification of decision makers.

ACKNOWLEDGMENTS

The authors would like to thank Ann Francis and Daniel Deavours for reading and commenting on previous versions of this paper. We would also like to acknowledge the support of EDS, an HP company, one of our partners on this project. Finally, we like to thank Larry Sackman of EDS, an HP company, for assisting with the short-haul rail trial.

REFERENCES

- [1] Federal Bureau of Investigation. (2006, July 21) Cargo Theft's High Cost. Headline. Federal Bureau of Investigation. [Online]. Available: http://www.fbi.gov/page2/july06/cargo_theft072106.htm
- [2] European Conference of Ministers of Transport, *Container Transport Security Across Modes*. Paris, France: Organisation for Economic Co-operation and Development, 2005.
- [3] OASIS. (2006, Oct 12) Reference Model for Service Oriented Architecture 1.0. OASIS Standard. [Online]. Available: <http://docs.oasis-open.org/soa-rm/v1.0/soa-rm.pdf>

- [4] KC SmartPort. (2008, Nov 10) Trade Data Exchange—Nothing short of a logistics revolution. Digital magazine. [Online]. Available: <http://www.joc-digital.com/joc/20081110/?pg=29>
- [5] J. Martin *et al.*, “Web services: Promises and compromises,” *Queue*, vol. 1, no. 1, pp. 48–58, Mar 2003.
- [6] H. Saiedian and S. Mulkey, “Performance evaluation of eventing web services in real-time applications,” *Communications Magazine, IEEE*, vol. 46, no. 3, pp. 106–111, Mar 2008.
- [7] J. Brown *et al.*, “SMS: The Short Message Service,” *Computer*, vol. 40, no. 12, pp. 106–110, Dec. 2007.
- [8] The Apache Software Foundation. (2008, Aug 24) Apache Axis2. Project documentation. The Apache Software Foundation. [Online]. Available: <http://ws.apache.org/axis2/>
- [9] OpenUDDI. (2008, Mar 7) Open UDDI. Project webpage. [Online]. Available: <http://openuddi.sourceforge.net/>
- [10] D. Griffin and D. Pesch, “A Survey on Web Services in Telecommunications,” *Communications Magazine, IEEE*, vol. 45, no. 7, pp. 28–35, July 2007.
- [11] Hi-G-Tek. (2009, Mar 17) Hi-G-Tek—Company. Corporate website. Hi-G-Tek. [Online]. Available: <http://www.higtek.com/>
- [12] D. Mulvey, “HSPA,” *Communications Engineer*, vol. 5, no. 1, pp. 38–41, February-March 2007.
- [13] C. E. Fossa *et al.*, “An overview of the IRIDIUM (R) low Earth orbit (LEO) satellite system,” in *Proc. IEEE 1998 National Aerospace and Electronics Conference, (NAECON 1998)*, Dayton, OH, USA, Jul 1998, pp. 152–159.
- [14] The Apache Software Foundation. (2007, Sep 1) Apache log4j. Project documentation. The Apache Software Foundation. [Online]. Available: <http://logging.apache.org/log4j/>
- [15] EsperTech. (2009, Feb 11) Esper – Complex Event Processing. Project documentation. EsperTech. [Online]. Available: <http://esper.codehaus.org/>
- [16] Google. (2009, May 6) Google Maps. Web mapping service. [Online]. Available: <http://maps.google.com>
- [17] D. L. Mills, “Internet Time Synchronization: the Network Time Protocol,” *Communications, IEEE Transactions on*, vol. 39, no. 10, pp. 1482–1493, Oct 1991.
- [18] D. Box *et al.* (2004, Aug 10) Web Services Addressing (WS-Addressing). Member submission. W3C. [Online]. Available: <http://www.w3.org/Submission/ws-addressing/>
- [19] J. Clark and S. DeRose. (1999, Nov 16) XML Path Language (XPath). W3C Recommendation. W3C. [Online]. Available: <http://www.w3.org/TR/xpath>
- [20] D. T. Fokum, “Optimal Communications Systems and Network Design for Cargo Monitoring,” To appear in Proc. Tenth Workshop Mobile Computing Systems and Applications (HOTMOBILE 2009). Santa Cruz, CA: ACM Press, Feb 2009.
- [21] Y.-C. Lai *et al.*, “Optimizing the Aerodynamic Efficiency of Intermodal Freight Trains,” *Transportation Research Part E: Logistics and Transportation Review*, vol. 44, no. 5, pp. 820–834, Sep 2008.



Technical Report

Summary of Status:
A Unified Architecture for SensorNet with Multiple
Owners: Supplement to Advance SensorNet
Technologies to Monitor Trusted Corridors

University of Kansas, ITTC
V.S. Frost, G.J. Minden, J.B. Evans, L. Searl,
D.T. Fokum, D. Deavours, E. Komp, A. Oguna,
M. Zeets, M. Kuehnhausen, D. Depardo

EDS
J. Walther, L. Sackman, M. Gatewood,
J. Spector, S. Hill, J. Strand

ITTC-FY2010-TR-41420-12

December 2008

Project Sponsor:
Oak Ridge National Laboratory
Award Number 4000043403

Abstract

This effort is aimed at monitoring cargo movements along a trusted corridor, e.g., rail facilities, in association with an integrated data-oriented methodology to increase efficiency and security. This goal is being achieved by performing research and deployment of an associated testbed focused on rail transportation issues. The results of this effort will lay the foundation for enhancing the ability of the private sector to efficiently embed security that provides business value such as safety, faster transport and reduced theft while supporting law enforcement and national security. In the end, the benefit of the combination of real-time sensor data with trade data exchange information will be demonstrated through field tests on a deployed rail testbed. (For background and definition of terms see [1] V.S. Frost, G.J. Minden, J.B. Evans, L. Searl and D.T. Fokum, T. Terrell, L. Sackman, M. Gatewood, J. Spector, S. Hill, and J. Strand, “Status Update : A Unified Architecture for SensorNet with Multiple Owners: Supplement to Advance Sensor Technologies to Monitor Trusted Corridors”, ITTC-FY2009-TR-41420-10 August 2008).

Table of Contents

Abstract	i
Table of Contents	ii
List of Figures	ii
List of Tables	ii
1.0 Introduction.....	1
2.0 Status on Technology Proof of Concept and Integration of the SmartPort Trade Data Exchange and Transportation Security SensorNet Technologies	1
3.0 Status of the Development of Transportation Security SensorNet (TSSN) Technologies	2
4.0 Status of System Architecture, Modeling, and Optimization	2
5.0 Status of Communications System Evaluation	3
6.0 Status RFID Technology Evaluation and Development	3
7.0 Associated Efforts	4
8.0 Project Timeline.....	4
9.0 References.....	5

List of Figures

Figure 1: Project Timeline	4
----------------------------------	---

List of Tables

Table 1: Example results from system trade-off study	3
--	---

1.0 Introduction

This effort is aimed at monitoring cargo movements along a trusted corridor, e.g., rail facilities, in association with an integrated data-oriented methodology to increase efficiency and security. This goal is being achieved by performing research and deployment of an associated testbed focused on rail transportation issues. The results of this effort will lay the foundation for enhancing the ability of the private sector to efficiently embed security that provides business value such as safety, faster transport and reduced theft while supporting law enforcement and national security. In the end, the benefit of the combination of real-time sensor data with trade data exchange information will be demonstrated through field tests on a deployed rail testbed. (For background and definition of terms see [1] V.S. Frost, G.J. Minden, J.B. Evans, L. Searl and D.T. Fokum, T. Terrell, L. Sackman, M. Gatewood, J. Spector, S. Hill, and J. Strand, “Status Update : A Unified Architecture for SensorNet with Multiple Owners: Supplement to Advance Sensor Technologies to Monitor Trusted Corridors”, ITTC-FY2009-TR-41420-10 August 2008).

2.0 Status on Technology Proof of Concept and Integration of the SmartPort Trade Data Exchange and Transportation Security SensorNet Technologies

In preparation for rail trials of the integration of the Smartport trade data exchange and transportation security sensornet technologies, trains at a rail yard in Kansas City, Mo. were visited on August 27, 2008. Initially a short haul rail trial will be conducted which will run from the rail yards in downtown Kansas City, Mo. to the intermodal facility in south Kansas City. Information gathered from this visit lead to the construction of the hardware required for the trail trials. The construction and testing of the required hardware has been completed.

A mobile integration test of the communications and interactions between the TDE at EDS, MRN, and the VNOC has been successfully completed (December 4, 2008). This mobile integration test was conducted using two pickup trucks to emulate the train. The mobile component (the MRN) of the integration test was conducted in driving around Lawrence, Ks.; the VNOC was located in ITTC on the KU campus, and the TDE was located at the EDS facilities in Overland Park, Ks.. This successful mobile integration test was preceded by several field experiments of components of the system.

With the successful completion of the mobile integration test, the short haul rail test is being scheduled with a target of before the end of 2008.

3.0 Status of the Development of Transportation Security SensorNet (TSSN) Technologies

The development of the TSSN takes an SOA approach, building upon the original ideas of ACE but utilizing current technology and widely accepted open Web Service specifications and publicly available implementations which are suitable for Sensor Networks. Some of the Web Service specifications in use are SOAP, the WS-X specifications, and UDDIv3.

The TSSN is being implemented in three phases. The first phase will be used in the field trials described above.

Phase 1 – Simple service messages based on OGC specifications (used in trials).

Phase 2 – Use full OGC specification interface messages.

Phase 3 – Use lessons learned from Phases 1 and 2 to make improvements.

Phase 1 is now complete.

4.0 Status of System Architecture, Modeling, and Optimization

This task is focused on developing models of the Transportation Security SensorNet (TSSN) and Trade Data Exchange environment that can be used to articulate trade-offs and enable system optimization. In order to model the container placement and sensor assignment problem efficiently a new method has been devised for indexing the containers and the locations (slots) that they occupy on the train as well as the location of sensors and elements of the communication network. We developed a new concept of object visibility and defined a visibility space as the set of system costs such that customer requirements for probability of detection, probability of false alarm and event reporting deadlines are met. The problem can now be formally stated as: Given a collection of objects with different values and end-to-end information systems (including sensors, seals, readers, and networks) with different capabilities: how do we design a system that allows “visibility” (meeting given constraints) while minimizing overall system cost? Small train based and trackside systems have been analyzed to confirm our approach. Sample results are given below (Table 1). (A full description of the current system model is in [2] Daniel T. Fokum, “Optimal Communications Systems and Network Design for Cargo Monitoring” Proposal for Ph.D. dissertation research Department of Electrical Engineering & Computer Science, University of Kansas, December 2008.)

Case	Number of Sensors	Normalized Cost Metric	Average Time to Record Event/s
Rail-mounted Scenario	5	1,555	6
	4	1,581	49,685
	3	3,145	99,363
	2	4,795	149,042
	1	7,300	198,721
	0	11,400	248,400
Trackside Case	5	1,895	556

Table 1 Example results from system trade-off study

Further work is needed on the model, objective function, and obtaining realistic model parameters. The framework will then be applied to study system trade-offs.

5.0 Status of Communications System Evaluation

Research is continuing on radio technologies for TSSN. As part of evaluating the current active container seal technology, it was discovered that the communication range for the devices selected for this research was more limited than expected. The active seals we are using operate in the 916 MHz band. A vendor of bidirection RF amplifiers in the 916 MHz band made custom modifications to their device based on our specification. With those modifications we were able to expand the communications range of the system. In the course of conducting the mobile integration tests we determined that the communications range is now on the order of a quarter of a mile. This expanded range will enhance the rail field tests. Note with all the elements (MRN, VNOC, and TDE) of the system in operation in a mobile environment exact range measurements are difficult to obtain.

6.0 Status RFID Technology Evaluation and Development

The combination of the new ITTC/KU on-metal RFID tag technology and the Mojix system was deployed and tested in a warehouse environment. While this initial testing focused on the suitability of the system on an MES (manufacturing and execution system, i.e., an assembly line) and for scanning entering and exiting a dock door, the results of this testing lead to conclusions concerning applicability in an intermodal environment. Additional experiments have been conducted and a technical report is in preparation.

7.0 Associated Efforts

KC SmartPort has continued to coordinate meetings for the groups involved in TSSN, CTIP and EFM. These meetings are creating a common, open environment with low entry barriers to enable broader access by stakeholders while contributing a venue to commercialization. The KU/ITTC and EDS teams are supporting the interactions between these efforts. KU/ITTC and EDS teams participated in KC SmartPort coordination meetings on August 27, September 30 and October 28, 2008. The next meeting is scheduled for December 16, 2008.

8.0 Project Timeline

Figure 1 is the current project timeline. The short haul field trial is targeted for completion by the end of 2008; a long haul field trial in Mexico is anticipated in spring 2009. The efforts associated with the system modeling, communications, and RFID are planned to be completed by the end of spring 2009 and an interim report describing these activities delivered by end of summer 2009. Activities associated with SmartPort, EFM, and CTIP will continue until June 2010. The current date of completion for the effort is June 15, 2010.

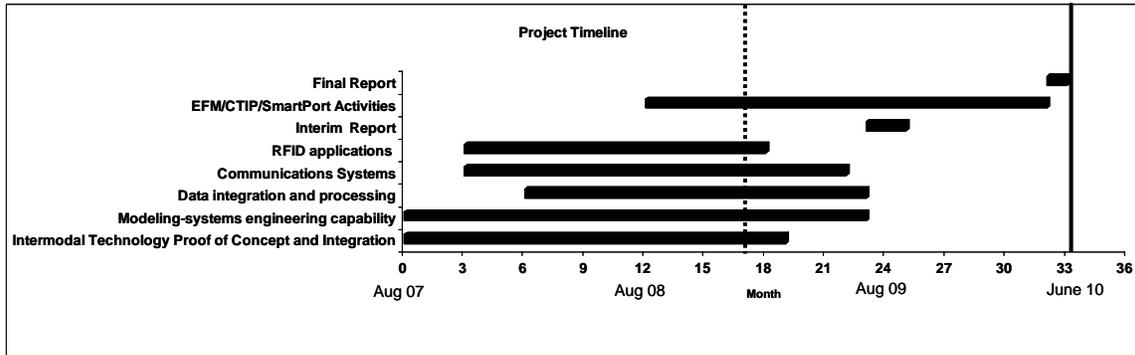


Figure 1 Project Timeline

9.0 References

- [1] V.S. Frost, G.J. Minden, J.B. Evans, L. Searl and D.T. Fokum, T. Terrell, L. Sackman, M. Gatewood, J. Spector, S. Hill, and J. Strand, “Status Update : A Unified Architecture for SensorNet with Multiple Owners: Supplement to Advance Sensor Technologies to Monitor Trusted Corridors”, ITTC-FY2009-TR-41420-10 August 2008.
- [2] Daniel T. Fokum “Optimal Communications Systems and Network Design for Cargo Monitoring”, Proposal for Ph.D. dissertation research, Department of Electrical Engineering & Computer Science, University of Kansas, December 2008.



Technical Report

Service Oriented Architecture for Monitoring Cargo in Motion Along Trusted Corridors

M. Kuehnhausen, D. T. Fokum, V. S. Frost,
D. DePardo, A. N. Oguna, L. S. Searl, E. Komp,
M. Zeets, D. D. Deavours, J. B. Evans,
and G. J. Minden

ITTC-FY2010-TR-41420-13

July 2009

Project Sponsor:
Oak Ridge National Laboratory

Abstract

This thesis describes a system called the *Transportation Security SensorNet* that can be used to perform extensive cargo monitoring. It is built as a *Service Oriented Architecture* (SOA) using open *web service* specifications and *Open Geospatial Consortium* (OGC) standards. This allows for compatibility, interoperability and integration with other *web services* and *Geographical Information Systems* (GIS).

The two main capabilities that the *Transportation Security SensorNet* provides are remote sensor management and alarm notification. The architecture and the design of its components are described throughout this thesis. Furthermore, the specifications used and the fundamental ideas behind a *Service Oriented Architecture* are explained in detail.

The system was evaluated in real world scenarios and performed as specified. The alarm notification performance throughout the system, from the initial detection at the *Sensor Node* service to the *Alarm Reporting* service, is on average 2.1 seconds. Location inquiries took 4.4 seconds on average. Note that the majority of the time, around 85% for most of the messages sent, is spent on the transmission of the message while the rest is used on processing inside the *web services*.

Finally the lessons learned are discussed as well as directions for future enhancements to the *Transportation Security SensorNet*, in particular to security, complex management and asynchronous communication.

Table of Contents

Abstract	i
Table of Contents	ii
List of Figures	vi
List of Listings	viii
List of Tables	ix
1 Introduction	1
2 Statement of Problem	3
2.1 Proprietary Solutions	3
2.2 Variety of Open Standards	6
2.3 Service Oriented Architecture	9
2.4 Summary	12
3 Background	13
3.1 Extensible Markup Language	13
3.1.1 Overview	14
3.1.2 Descriptive power	19
3.1.3 Ease of transformation	19
3.1.4 Information storage and retrieval	21
3.1.5 Flexible transmission	22
3.2 Open Geospatial Consortium	22
3.2.1 Sensor Web Enablement (SWE)	24
3.2.2 Geography Markup Language (GML)	26
3.2.3 Catalogue Service for Web (CSW)	28
3.2.4 Observations & Measurements (O&M)	29

3.2.5	Sensor Observation Service (SOS)	31
3.2.6	Sensor Alert Service (SAS)	33
4	Service Oriented Architecture	36
4.1	Representational State Transfer (REST)	39
4.1.1	Traditional Definition	40
4.1.2	Current Use	40
4.1.3	Further Development	42
4.2	SOAP	42
4.2.1	Message format	43
4.2.2	Faults	44
4.2.3	Further development	45
4.3	Web Service Specifications	46
4.3.1	WS-Addressing	46
4.3.2	WS-Eventing	47
4.3.3	WS-Security	48
4.4	Service Directory	51
4.5	Web Services Description Language (WSDL)	53
4.5.1	Description	55
4.5.2	Types	56
4.5.3	Interface	57
4.5.4	Binding	57
4.5.5	Service	58
4.6	Message Exchange Patterns	58
4.6.1	In-Only	59
4.6.2	Robust In-Only	60
4.6.3	In-Out	60
4.6.4	In-Optional-Out	61
4.6.5	Out-Only	61
4.6.6	Robust Out-Only	62
4.6.7	Out-In	62
4.6.8	Out-Optional-In	62
5	Related Work	64
5.1	Microsoft - An Introduction to Web Service Architecture	64
5.2	Adobe - Service Oriented Architecture	66
5.2.1	Request-Response via Service Registry (or Directory)	67
5.2.2	Subscribe-Push	67

5.2.3	Probe and Match	68
5.3	Open Sensor Web Architecture	69
5.4	Globus - Open Grid Services Architecture	71
5.5	Service Architectures for Distributed Geoprocessing	74
5.6	Web Services Orchestration	77
5.7	Summary	78
6	Design & Architecture	80
6.1	Overview	80
6.1.1	Service Oriented Architecture	80
6.1.2	Services	88
6.1.3	Clients	89
6.1.4	Modules	90
6.1.5	Subscriptions	92
6.1.6	Synchronous and asynchronous communication	92
6.2	TSSN Common Namespace	94
6.3	Mobile Rail Network	97
6.3.1	Sensor Node	97
6.3.2	Alarm Processor	102
6.4	Virtual Network Operation Center	104
6.4.1	Sensor Management	105
6.4.2	Alarm Processor	109
6.4.3	Alarm Reporting	111
6.5	Trade Data Exchange	119
6.5.1	Trade Data Exchange Service	119
6.6	Open Geospatial Consortium Specifications	122
7	Implementation Results	123
7.1	Logging Module	123
7.2	Log Parser	124
7.2.1	Abstraction Layer Model	124
7.2.2	Message Types	125
7.3	Visualization	127
7.4	Performance and Statistics	128
7.4.1	Road Tests with Trucks	128
7.4.2	Short Haul Rail Trial	130

8 Conclusion	134
8.1 Current Implementation	134
8.2 Future work	135
8.3 Acknowledgment	137
References	138

List of Figures

3.1	OGC standardization framework as described in [74]	23
3.2	Sensor Web Concept from [10]	25
3.3	Catalogue Service reference model architecture from [63]	28
3.4	Observation process as described in [20]	29
3.5	SOS data publishing process as described in [60]	32
3.6	SOS data consumption process as described in [60]	32
3.7	SAS advertising process described in [78]	33
3.8	SAS notification process described in [78]	34
4.1	Service overview	36
4.2	Traditional web applications and AJAX from Garrett [35]	41
4.3	SOAP message format	44
4.4	WSDL 2.0 overview	54
4.5	In-Only message exchange pattern	59
4.6	Robust In-Only message exchange pattern	60
4.7	In-Out message exchange pattern	60
4.8	In-Optional-Out message exchange pattern	61
4.9	Out-Only message exchange pattern	61
4.10	Robust Out-only message exchange pattern	62
4.11	Out-In message exchange pattern	62
4.12	Out-Optional-In message exchange pattern	63
5.1	Request-Response via Service Registry (or Directory) message exchange pattern from [65]	67
5.2	Subscribe-Push message exchange pattern from [65]	68
5.3	Probe and Match message exchange pattern from [65]	68
5.4	NOSA layer overview from [19]	69
5.5	Globus Toolkit overview from http://www.globus.org/toolkit/about.html	72

5.6	Forest fire application from [34]	75
5.7	Forest fire web services architecture from [34]	75
5.8	Web orchestration framework from [47]	78
6.1	Service message overview	81
6.2	Service cloud	82
6.3	Axis2 extensibility from [16]	84
6.4	Axis2 modules from [16]	85
6.5	Service composition	87
6.6	Mobile Rail Network message overview	97
6.7	Mobile Rail Network Sensor Node	98
6.8	Mobile Rail Network Alarm Processor	102
6.9	Virtual Network Operation Center message overview	105
6.10	Virtual Network Operation Center Sensor Management	106
6.11	Virtual Network Operation Center Alarm Processor	109
6.12	Esper architecture from [27]	109
6.13	Virtual Network Operation Center Alarm Reporting	111
6.14	Trade Data Exchange message overview	119
6.15	Trade Data Exchange Service	120
7.1	SOAP message (left) to Log parser classes (right) comparison	124
7.2	Two transmit-receive pairs (red and green)	126
7.3	A message couple (red)	126
7.4	Log file and service interaction visualization	127
7.5	Request performance from [31]	131
7.6	Network transmission and processing performance from [31]	131
7.7	System alarm notification performance from [31]	132

List of Code Listings

3.1	Simple XML book description	14
3.2	Library of books	15
3.3	Library of books using attributes	15
3.4	Extended library of books	16
3.5	Element book format	16
3.6	Element book format with type (elementBook.xsd)	17
3.7	Attribute book format with type (attributeBook.xsd)	18
3.8	Library schema (library.xsd)	18
3.9	Library stylesheet (library.xsl)	20
3.10	Library of books in HTML (library.html)	20
4.1	SOAP message format example	44
4.2	SOAP Fault message example	44
4.3	WSDL Description example	55
4.4	WSDL Types example	56
4.5	WSDL Interface example	57
4.6	WSDL Binding example	57
4.7	WSDL Service example	58

List of Tables

3.1	Example XPath expressions	21
3.2	Collection types from [20]	30
6.1	Sensor Node StartMonitorSensors operation	99
6.2	Sensor Node StopMonitorSensors operation	99
6.3	Sensor Node setSensors operation	99
6.4	Sensor Node AddSeals operation	100
6.5	Sensor Node getLocation operation	100
6.6	Sensor Node GetCapabilities operation	101
6.7	Sensor Node GetObservation operation	101
6.8	Alarm Processor Alert operation	103
6.9	Alarm Processor SensorNodeEvent operation	103
6.10	Alarm Processor SetMonitoringState operation	104
6.11	Sensor Management startMonitoring operation	106
6.12	Sensor Management stopMonitoring operation	107
6.13	Sensor Management getLocation operation	107
6.14	Sensor Management setAlarmSecure operation	108
6.15	Sensor Management setAlarmProcessorMonitoringState operation	108
6.16	Alarm Processor MRN_Alarm operation	110
6.17	Alarm Reporting addSmsProvider operation	112
6.18	Alarm Reporting updateSmsProvider operation	113
6.19	Alarm Reporting removeSmsProvider operation	113
6.20	Alarm Reporting removeSmsProviderById operation	113
6.21	Alarm Reporting getAllSmsProviders operation	114
6.22	Alarm Reporting addContact operation	114
6.23	Alarm Reporting updateContact operation	115
6.24	Alarm Reporting removeContact operation	115
6.25	Alarm Reporting removeContactById operation	115
6.26	Alarm Reporting getAllContacts operation	116

6.27	Alarm Reporting addAlarmContactMapping operation	116
6.28	Alarm Reporting updateAlarmContactMapping operation	117
6.29	Alarm Reporting removeAlarmContactMapping operation	117
6.30	Alarm Reporting removeAlarmContactMappingById operation	117
6.31	Alarm Reporting getAllAlarmContactMappings operation	118
6.32	Alarm Reporting NOC_Alarm operation	118
6.33	Alarm Reporting getAllAlarms operation	118
6.34	TradeDataExchange ShipmentQuery operation	121
6.35	TradeDataExchange ValidatedAlarm operation	121
7.1	XPath expressions for <i>WS-Addressing</i>	125

Chapter 1

Introduction

Cargo theft and tampering are common problems in the transportation industry. According to Wolfe [85] the “FBI estimates cargo theft in the U.S. to be \$18 billion” and the Department of Transportation “estimated that the annual cargo loss in the U.S. might be \$20 billion to \$60 billion”. Wolfe [85] also gives good reason to believe that the actual number may be even higher than \$100 billion because of two reasons. First it is assumed that about 60 percent of all thefts go unreported and second the indirect costs associated with a loss are said to be three to five times the direct costs.

With the advances in technology, this problem has evolved into a cat-and-mouse game where thieves constantly try to outsmart the newest cutting edge security systems.

In terms of securing cargo, there are usually two aspects: first ensuring the physical safety of the cargo and second monitoring and tracking it. The latter especially has become of more interest as of late because many shipments cross national borders and cargo may be handled by a multitude of carriers. All of this leads to a huge demand for tracking and monitoring systems by the cargo owners, carriers, insurance companies, customs and many others.

In this thesis, a framework is introduced which builds on open standards and software components to allow “monitoring cargo in motion along trusted corridors”. The focus lies on the use of a *Service Oriented Architecture* and *Geographical Information System*

specifications in order to allow an industry wide adoption of this open framework.

A formal description of the problem to be analyzed can be found in chapter 2. In particular, it discusses the problems of proprietary systems, the advantages of open standards and the approach of using a *Service Oriented Architecture* in the transportation industry.

Chapter 3 gives an in-depth introduction to the *Extensible Markup Language* that is used as the foundation of the framework. Furthermore the specifications provided by the *Open Geospatial Consortium* that define the elements and interfaces for *Geographical Information Systems* are described.

The formal representation of the framework is a *Service Oriented Architecture* which is described in chapter 4 along with the components that it uses.

Chapter 5 refers to related work and focuses on the topics that either deal with the *Service Oriented Architecture* or the *Open Geospatial Consortium* specifications.

The main part of this thesis that details the design and architecture of the framework can be found in chapter 6. It explains the individual components as well as the software parts and specifications that are used in the implementation.

Chapter 7 gives test and performance results and describes the tools that have been developed for that particular purpose.

The thesis concludes with chapter 8 that also provides an outlook for future work.

Chapter 2

Statement of Problem

In order to address the problem of cargo theft, the *Transportation Security SensorNet* project has been created. Its goal is to promote the use of open standards and specifications in combination with web services to provide cargo monitoring capabilities.

The main question is the following:

“How can a *Service Oriented Architecture*, open standards and specifications be used to overcome the problems of proprietary systems that are currently in place and provide a reusable framework that can be implemented across the entire transportation industry?”

The three main aspects of this question are discussed next.

2.1 Proprietary Solutions

Current commercial systems in the transportation industry are often proprietary. This is because a lot of effort is spent on research and development in order to create what is called *intellectual property*. The assumption is then that as long as the competitors do not have access to the system and its protocols that *intellectual property* is safe and provides a competitive advantage. Another common “benefit” of keeping the

systems closed is the perceived additional security since in order to successfully attack the system its implementation and protocols have to be *reverse engineered*.

The problem with this is that these advantages are often one-sided and favor vendors. Once a proprietary system has been implemented it has to be maintained. What happens if a customer that uses the system invested a lot of money into a its infrastructure and the training of its employees and the company that provides the system releases a new version of it which of course costs money again. The customer has several choices:

Upgrade Throughout the literature this is often considered the most expensive option because of the cost for the upgrade to the new version and the additional training to the employees that has to be provided. The benefits of upgrading are the use of new technology, potential gains in efficiency through new features and the latest bug fixes.

Do Not Upgrade By many regarded as the most cost efficient solution, choosing not to upgrade compromises new features and updates for the ability to save costs. An approach that is taken by some companies is the so-called *skip a version* technique. This allows companies to plan better as internal processes and systems often have to interoperate and need to remain compatible to each other.

Change Vendor In this situation, the new version of the system that is provided by company A does not provide the necessary features or is simply too expensive. Furthermore, a different company B offers a similar product with more features or for less money. The move to the new system is now dependent on the following things: How big are the estimated savings and what are the direct and indirect costs of the transition? It often happens that after careful consideration the costs outweigh the estimated gains and the customer goes back to considering whether or not to simply upgrade. If a transition is made, the process could be time consuming and turn out to be more complicated than expected.

Picture this extreme case as well. What happens if the vendor goes out of business? All of the sudden, the short-term goal is to maintain support for the system and to keep it running while in the long-term to look for a suitable replacement and be forced to transition. Even if this case does not happen the dependency on the vendor can be crucial. If the system has errors or a particular enhancement is desperately needed, the vendor decides what to do about it. For big companies that are major customers this may not be such a big problem because they often get preferential treatment. But for small and medium businesses the wait might be too long and lose them customers and revenue.

The main point here is that many customers are locked into proprietary solutions that are incompatible with similar solutions offered by competitors. In a 2003 survey by the Delphi Group [36] it was found that 52% of developers and 42% of consumers see standards enabling the “approval of projects otherwise threatened by concerns over proprietary system lock-in”. Furthermore, an overwhelming 71% of developers and 65% of consumers feel that the use of open standards “increases the value of existing and future investments in information systems”.

The problem of non-interoperability with regard to geospatial processing is the topic of a paper by Reichardt [75]. Because *Geographical Information Systems* are often immensely complex, companies that invest heavily into this area often only support their product. As described in the sample scenario, this leads to a lack of coordination among entities such as the *Federal Emergency Management Agency* (FEMA), the *National Transportation Safety Board* (NTSB) and the *Environmental Protection Agency* (EPA) because of the inability to share vital information which is the key to fast decision making and data analysis

2.2 Variety of Open Standards

The idea of open standards and specifications is to define so-called *interfaces* and *protocols* that can be used as references for the implementation of a system. There are many standards committees and industry groups that aim to define them, most often focused on a particular area. Some of the most well-known ones include the *World Wide Web consortium* (W3C), the *Organization for the Advancement of Structured Information Standards* (OASIS), the *International Telecommunication Union* (ITU) and the *International Organization for Standardization* (ISO).

The main principles that govern the development of standards are usually the same across all organizations. The following is an overview according to ISO:

Consensus All parties that are affected by the proposed standard get the chance to voice their opinions. This includes initial ideas and continues with feedback and comments during the standardization process.

Industrywide The idea is to develop global standards that can be used worldwide by entire industries.

Voluntary The standardization process is driven by the people that are interested in it and that see its future benefits across a particular industry. It is often based on so-called *best practices* that are already commonly in use.

The importance of open standards is emphasized in a paper by McKee [56]. It provides the evolution and success of the Internet as the “perfect example” for the use of open standards. In particular it explains that since the Internet is based upon communication and communication means “transmitting or exchanging through a *common system* of symbols, signs or behavior”, the process of standardization can basically be seen as “agreeing on a *common system*”. The other parts of the paper are focused on

how so-called *openness* can help *Geographical Information Systems* (GIS) but many of the points mentioned apply to open standards in general.

In particular the following aspects are associated with open standards:

Compatibility This includes the ability to share data across vendors and systems in a uniform and non-proprietary form. It allows processes to use essentially the same data in order to perform their specific task without the need of costly conversions or interpretation errors. Most *common* formats are also backward compatible which means that no particular version of the system is needed to interpret the data. Only a certain subset of functionality might be provided when using in older versions though. Another advantage of open formats is the fact that even if a particular version of a format is completely outdated and only used in legacy systems, its specification is still accessible to everyone. Hence systems can still be designed to use the format.

Freedom of Choice A major problem of proprietary solutions that was described earlier was the so-called *vendor lock*. Once a customer implements a proprietary system and builds its infrastructure around it, choices in the future are limited. Open standards by definition are vendor independent. Furthermore many of them support a broad variety of implementation scenarios. These implementations often are not even limited to a particular platform, operation system or programming language. This is especially true for most of the web standards.

Interoperability Through the use of clearly defined interfaces, standards dramatically enhance interoperability. The standards that define interface specifications do not provide a specific implementation but provide references to *best practices* and *implementation patterns* instead. Companies choose what kind of system implementation they prefer. This allows them to make use of existing infrastructure and capabilities that might otherwise have to be changed when using a proprietary system. The uniform access to functionality and data enables companies to connect a multitude of systems

and make more use of them. Also, in case one part of the system has to be replaced, another one that simply provides the same interface can take its place. This allows great flexibility in terms of the overall system design.

Leverage For companies the standardization of concepts, frameworks and common approaches provides a number of benefits. Since research and development can be extremely cost intensive, companies want to make sure there is a guaranteed *return on investment* for them. Open standards do not necessarily lead to increased revenue but they do provide insurance to the companies that they are on the “right” track and what they implement is actually used industrywide. This is very important because customers are aware that when they purchase a system from company A that uses a proprietary or non-standard implementation they might become a victim of *vendor lock*. Acquiring a system that is build on open standards allows them to choose the best and most cost effective solution from a variety of independent implementations. Another advantage is that once different implementations by the main vendors have been established, there is room for custom solutions by smaller vendors, often in the form of extensions or plugins.

Open Source The biggest benefit of using open standards is that fact it leads to innovation. This is because everybody can contribute, suggest enhancements, outline *best practices* and address mistakes. In terms of software this approach is often referred to as *open source*.

However, there are several problems that can be associated with non-proprietary systems. Implementations are based upon the interpretation of the standards which may differ significantly. Furthermore, some implementations only support a subset of the original specification, are slower than the reference implementation or use incompatible sub systems.

2.3 Service Oriented Architecture

The concept of information processing and sharing across various applications using so-called *web services* is the main focus of this thesis. The basic idea is to define components of a system as *services* and users as *clients* that can retrieve data from them. Note that interaction between *services* is done using so-called *embedded clients*. The *services* take care of things such as information processing, data analysis and storage. With all business logic embedded into *services* and interaction between them clearly defined using open standards an infrastructure is built that is called the *Service Oriented Architecture* (SOA).

The Internet allows the following two things that are relevant to geospatial processing: a common means of communication and the ability for efficient information sharing. There exist many standards on how to transmit, receive, encode and decode data. SOA builds on top of them to provide new specifications that enable the design, implementation and use of *web services*. Through these *web services* companies, government agencies and others have the ability to share and process information in a uniform manner which cuts costs, time and resources and improves efficiency. More information on the *Service Oriented Architecture* can be found in chapter 4.

Now why is the SOA such an “enabler”? What is possible now that was not possible before? According to Irmen [44] *automation* and *efficient communication* with partners are the two most important things in *supply chain management* which represents the core of the transportation industry. Let us take a look at how the *Service Oriented Architecture* addresses both of them in regard to the individual topics outlined in the paper.

Automation A vital part in transportation is the so-called *screening* process. Companies that transport goods must ensure safety and therefore check all parties involved in the trade. An important aspect of this is the use of a so-called *denied trade list*

which lists items and companies that are not allowed to import or export into specific countries. With the reduction in manual labor and transition to a *web services* based system that automatically performs these checks, efficiency could be greatly increased.

A closely related topic is *accountability*. Who is responsible if something goes wrong during the trade process? Since goods are often handled by many different parties, it must be possible to monitor the location of cargo and handovers tightly. This is especially important in cases of tampering or even theft of the cargo.

Furthermore, agencies and customs more and more require *electronic trade information* instead of paper documents in order to track trade. Because of different formats and legacy applications that are often unable to provide this information in its entirety, additional resources have to be allocated in order to remain compliant with current practices. *Web services* and open standards can overcome this problem with uniform interfaces and common data formats.

Having the ability to *monitor* the location not just for perishable goods but also for high value goods is of great importance in the transport chain. Current processes should be able to automatically route cargo based on its needs and cost effectiveness.

Irmen [44] also points out that “the lack of integration between products causes users to deal with multiple systems having disparate data and non-uniform input and output” and calls for the use of a single platform. Using the *Service Oriented Architecture* this “call” becomes less necessary because it is platform independent and at the same time able to provide integration of multiple systems and standardized data formats.

Efficient Communication Building a virtual network among the parties involved in the trade process establishes efficient means of communication. It allows the *coordination* between otherwise disparate entities that is essential to provide cost effective and reliable shipping of cargo. The Internet provides the communication layer but it is the standards of *web services* that enable the integration of different systems.

Irmen [44] mentions the so-called *Software-as-a-Service* (SaaS) approach which allows software to run on a per-use basis without the costs of complex hardware infrastructure. This works very well with SOA as the interfaces defined by those services are often *web services* interfaces that are essentially part of SOA.

Security within the transportation industry plays a big role because trade data is to be kept confidential at all times and only distributed on a need-to-known basis. This puts an additional burden on the parties that are involved, as the parties must exchange data confidentially at each point of interaction. If open standards are used for this, *security* is implemented based on interfaces and policies that are easy to manage.

In order to manage the transportation chain in its entirety, a *global view* is often needed. This is problematic since individual parties often only deal with their respective neighbors. Using open standards and the *Service Oriented Architecture* approach each party could provide an uniform information interface that is accessible to other parties in the chain. This allows consistent reporting, monitoring and analysis at each step during the shipping process.

The reporting part especially has gained more attention over the past years as the focus has shifted towards more ethical and socially responsible business practices. *Accountability* coincides with this *social visibility* and therefore improvements in monitoring cargo not only lead to increased revenue on the business side but better public relations as well.

Overall the paper by Irmen [44] gives excellent reasons for open systems in terms of accountability, coordination, scalability and cost. It outlines important aspects that need to be taken into consideration when designing an architecture such as the *Transportation Security SensorNet*.

2.4 Summary

The following chapters describe how open specifications for *Geographical Information Systems* in combination with *web services* can be used to address the problems of proprietary systems that were outlined in section 2.1. In the *Transportation Security SensorNet* (TSSN) this is achieved by using a variety of open standards primarily because of the aforementioned *interoperability* and *freedom of choice* (see section 2.2). The use of a *Service Oriented Architecture* for the TSSN allows the creation of the applications needed for efficient and cost effective transportation chains (see section 2.3).

Chapter 3

Background

3.1 Extensible Markup Language

The *Extensible Markup Language* (XML) is a specification by the *World Wide Web Consortium* (W3C) that is used to describe data in a highly flexible but also concise way. It serves as the basis for most of the specifications that are referenced in this thesis.

As described by Sperberg-McQueen et al. [81] one of the main goals of the specification is interoperability and support for a multitude of applications. This is emphasized by the fact that XML should be human-readable and easy to process by computers. XML can be used to describe, filter and format data while providing storage functionality as well.

In the *Transportation Security SensorNet* it is utilized in a variety of ways. The *web services* and the *Open Geospatial Consortium* standards define their interfaces and data elements using XML. SOAP, as described in section 4.2, is a XML message format that is used as the basis for the transmission of data in the framework. Furthermore, many configuration files for the web services and clients in the *Transportation Security SensorNet* are in XML. The use of the *Extensible Markup Language* is one of the main reasons for the flexibility and reusability of the framework

3.1.1 Overview

In the following sections some basic principles of XML are introduced. Let us start by describing a simple book using XML.

```
1 <?xml version="1.0" encoding="UTF-8" ?>
2 <book>
3   <!-- english title -->
4   <title>Hamlet</title>
5   <!-- author name -->
6   <author>William Shakespeare</author>
7 </book>
```

Listing 3.1 Simple XML book description

The first line is the XML declaration. It specifies that the described document uses version 1.0 of the XML specification and UTF-8 encoding. Line two starts with the definition of a book that contains a title (line four) and an author (line six). Note that line three and five are comments that are not part of the actual data but can be used to further describe it to humans. This example shows that XML can be as descriptive to humans as it is to computers.

Looking at the XML we can see multiple things. The element *book* has a so-called *start-tag* (line two) and an *end-tag* (line seven). Information about the specific book is kept in between these tags. As for the title and author information the actual data is also contained within their *start-tag* and *end-tags*. This demonstrates one basic type that is used most frequently in XML, an *element*. An *element* consists of a *start-tag* and an *end-tag* with either content or other *elements* in between. Note that there are also so-called *empty-element-tags* that look like `<empty-element/>`. They contain no further content or *elements*.

One of the requirements of using XML in applications is that one needs to define one specific *root element*. Therefore if we wanted to define more books let us put them into a library *root element*.

```

1 <?xml version="1.0" encoding="UTF-8" ?>
2 <library>
3   <book>
4     <title>Hamlet</title>
5     <author>William Shakespeare</author>
6   </book>
7   <book>
8     <title>Great Expectations</title>
9     <author>Charles Dickens</author>
10  </book>
11  ...
12 </library>

```

Listing 3.2 Library of books

XML is flexible enough to use different descriptions of essentially the same data. The following example represents the same library using *attributes* for title and author information instead of *elements*. *Attributes* are basically *name-value pairs* that contain information about the *element* that they are a part of.

```

1 <?xml version="1.0" encoding="UTF-8" ?>
2 <library>
3   <book title="Hamlet" author="William Shakespeare" />
4   <book title="Great Expectations" author="Charles Dickens"
5     />
6   ...
7 </library>

```

Listing 3.3 Library of books using attributes

The “problem” with this is that if one application uses *elements* and the other application uses *attributes* to describe books in their libraries they seem incompatible. In order to solve this we need to make sure that each description is uniquely identifiable. This can be done declaring so-called *namespaces* as described by Bray et al. [13]. The idea is to attach a specific *Uniform Resource Identifier* (URI) (see Berners-Lee et al. [6]) to the document or element definitions. For example, this would result in `<a:book xmlns:a="http://www.sample.com/elementBook">` for listing 3.2 and `<b:book xmlns:b="http://www.sample.com/attributeBook">` for listing 3.3. Using

these *namespaces* we have the ability to mix different descriptions in a single document.

```
1 <?xml version="1.0" encoding="UTF-8" ?>
2 <library xmlns="http://www.sample.com/library">
3   <a:book xmlns:a="http://www.sample.com/elementBook">
4     <a:title>Hamlet</a:title>
5     <a:author>William Shakespeare</a:author>
6   </a:book>
7   <b:book xmlns:b="http://www.sample.com/attributeBook"
8     title="Great Expectations" author="Charles Dickens" />
9   ...
10 </library>
```

Listing 3.4 Extended library of books

We can also use namespaces to uniquely identify document descriptions. The default description in listing 3.4 is `<library xmlns="http://www.sample.com/library">` and more specific descriptions are in place for each book.

So what do these descriptions actually look like? They are written in XML as well and called *XML Schema Definitions* (XSD). An overview is provided by Fallside and Walmsley [28] and the exact structure by Mendelsohn et al. [57]. While there are other standards in place for describing XML documents, *XML schemas* are the most common.

Let us describe the first book format.

```
1 <?xml version="1.0" encoding="UTF-8" ?>
2 <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
3   targetNamespace="http://www.sample.com/elementBook"
4   xmlns="http://www.sample.com/elementBook">
5   <xsd:element name="book">
6     <xsd:complexType>
7       <xsd:sequence>
8         <xs:element name="title" type="xsd:string"/>
9         <xs:element name="author" type="xsd:string"/>
10      </xsd:sequence>
11    </xsd:complexType>
12  </xsd:element>
13 </xsd:schema>
```

Listing 3.5 Element book format

We defined an element called *book* that contains two elements called *title* and *author*. Both of them are of type *string* which is a predefined data type. For ease of use and compatibility reasons the specification defines a set of standard data types. The type of *book* is so-called *complex* since it is the parent of other elements. Because this type is defined implicitly it is called *anonymous typing*. If one wanted to reuse the *book* type in some other element definition it makes more sense create a *complex book* type and define an *element* that is of this *type*. The XML schema would then take the following form:

```
1 <?xml version="1.0" encoding="UTF-8" ?>
2 <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
3   targetNamespace="http://www.sample.com/elementBook"
4   xmlns="http://www.sample.com/elementBook">
5   <xsd:element name="book" type="BookType"/>
6   <xsd:complexType name="BookType">
7     <xsd:sequence>
8       <xs:element name="title" type="xsd:string"/>
9       <xs:element name="author" type="xsd:string"/>
10    </xsd:sequence>
11  </xsd:complexType>
12 </xsd:schema>
```

Listing 3.6 Element book format with type (elementBook.xsd)

Line three defines the so-called *target namespace* of the schema. When the schema is used in a document, elements from it will automatically have this namespace. Line four specifies the *default namespace* for the schema so that elements and types in the schema are able to reference each other. The *sequence* tag at line seven specifies that the elements are to be in order, first *title* and then *author*. Other common options include *all* for random order and *choice* for the exclusive selection of elements.

The second book format could be defined by the following schema:

```

1 <?xml version="1.0" encoding="UTF-8" ?>
2 <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
3   targetNamespace="http://www.sample.com/attributeBook"
4   xmlns="http://www.sample.com/attributeBook">
5   <xsd:element name="book" type="BookType"/>
6   <xsd:complexType name="BookType">
7     <xsd:attribute name="title" type="xsd:string"/>
8     <xsd:attribute name="author" type="xsd:string"/>
9   </xsd:complexType>
10 </xsd:schema>

```

Listing 3.7 Attribute book format with type (attributeBook.xsd)

The only major difference in listing 3.7 is using an *attribute* instead of an *element* for the *book* information. Since our library should be able to use both descriptions let us define a schema that will allow this.

```

1 <?xml version="1.0" encoding="UTF-8" ?>
2 <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
3   xmlns:a="http://www.sample.com/elementBook"
4   xmlns:b="http://www.sample.com/attributeBook"
5   targetNamespace="http://www.sample.com/library"
6   xmlns="http://www.sample.com/library">
7   <xsd:import namespace="http://www.sample.com/elementBook"
8     schemaLocation="elementBook.xsd"/>
9   <xsd:import namespace="http://www.sample.com/
10     attributeBook" schemaLocation="attributeBook.xsd"/>
11   <xsd:element name="library">
12     <xsd:complexType>
13       <xsd:choice minOccurs="0" maxOccurs="unbounded">
14         <xs:element ref="a:book"/>
15         <xs:element ref="b:book"/>
16       </xsd:choice>
17     </xsd:complexType>
18   </xsd:element>
19 </xsd:schema>

```

Listing 3.8 Library schema (library.xsd)

The two previously defined schemas are imported in lines seven and eight. Line twelve and thirteen use so-called *references* to these defined elements. In this case we define the number of occurrences of each element explicitly. This is because by default

all elements have a `minOccurs=1` and a `maxOccurs=1`, meaning that they are required but may appear only exactly once. Hence, the library consists of books either in *element* or *attribute* format and the possible number of books ranges from *none* to *infinite*.

The examples that were covered illustrate how XML can be used to describe and store data. But what are the advantages of using XML over other technologies that can essentially do the same? One of the main reasons why the use of XML has grown in recent years is because of the impact of the *Internet*. Applications and data that were previously stored internally, often in proprietary formats, are now made accessible to *remote* locations and users. The need to deal with data in a more open and flexible way became apparent especially for web applications and services. The following sections describe the different ways of how web applications and applications in general can utilize and benefit from XML.

3.1.2 Descriptive power

The description of data using XML enables applications to be very flexible and modular. New fields or attributes of data can be added using schema extensions and applications can choose either to use the extension or the original XML schema definition. Data can even be entirely rearranged using new or modified element and type definitions. This allows different views of the same data which decreases conversion costs and increases reusability and interoperability. In essence the data stays the same, the only thing that changes is its interpretation.

This aspect is essential in a *Service Oriented Architecture* like the *Transportation Security SensorNet* because clients and web services are highly dynamic. Using XML allows the entire framework to be implemented in a flexible, modular and reusable way.

3.1.3 Ease of transformation

Data often needs to be transformed or converted from one format into the other. Since XML only describes the data we can transform it easily into whatever is needed.

For this reason *Extensible Stylesheet Language Transformations* (XSLT) as described by Kay [46] have been introduced. They enable automatic conversion of XML documents using so-called *stylesheets* that are defined in XML. Let us take the initial library in listing 3.2 and transform it into a simple HTML web page.

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/
   Transform" version="2.0" >
3   <xsl:template match="/">
4     <html>
5       <body>
6         <h1>Library books</h1>
7         <xsl:for-each select="library/book">
8           <div><xsl:value-of select="title"/> by <xsl:value-of
              select="author"/></div>
9         </xsl:for-each>
10        </body>
11      </html>
12    </xsl:template>
13  </xsl:stylesheet>
```

Listing 3.9 Library stylesheet (library.xml)

In listing 3.9 a header is specified that displays “Library books”. For each book in the library the title and author are then extracted and put into a relationship sentence. Hence, the resulting output would look like the following:

```
1 <html>
2   <body>
3     <h1>Library books</h1>
4     <div>Hamlet by William Shakespeare</div>
5     <div>Great Expectations by Charles Dickens</div>
6     ...
7   </body>
8 </html>
```

Listing 3.10 Library of books in HTML (library.html)

Note that this is just one of the many possibilities of converting an existing XML document into a different format. Within the *Transportation Security SensorNet* these *Extensible Stylesheet Language Transformations* are used by *Apache Axis2* to create

Java classes from *XML Schema Definitions* and *Web Services Description Language* files so that they can be used by clients and web services (see 6.1.1.2 and section 6.1.1.4).

3.1.4 Information storage and retrieval

Storing data in an XML format makes the data and relations between data more flexible. Databases often face the problem of *sparsity* where when a new column is added to a table all entries must have this new column. XML works in a different way. Additional information fields can be added just to the elements that need them while for all other elements the XML schema would simply define the field as *optional*. This can potentially save a lot of space when compared to storing the same data in traditional databases. In the *Transportation Security SensorNet* this “cost saving” approach is utilized by SOAP during the message transmission (see section 4.2).

In order to retrieve information efficiently from XML several specifications have been designed. Boag et al. [8] describes *XPath* which is a query language specifically designed for XML. It works on the basis of a document tree, the so-called *data model*, that it creates from the original XML. Elements are *nodes* in the tree and attributes so-called *attribute nodes*. Information can then be retrieved using *path expressions*. Table 3.1 shows some examples of the information that we are able to retrieve and the *path expressions* that were used for the library in listing 3.2. *XPath* is used by the *Log Parser* extract information from log files (see section 7.2).

XPath expression	Result
library	all books of the library
library/book[1]	first book
//author	all authors
//author/text()	all author names
//book[title="Hamlet"]/author/text()	author name of Hamlet

Table 3.1. Example XPath expressions

Another specification that is used for XML data information retrieval is called *XQuery* which was defined by Siméon et al. [77]. It is more complex and builds on

top of *XPath 2.0*. Immediate result computations and transformations are possible using a so-called *FLOWR* expressions. Where *Xpath* simply extracted information, *XQuery* enables applications and users to directly modify or change the appearance of the information.

3.1.5 Flexible transmission

Since there is a significant overhead associated with conversion, standards have been defined that allow various forms of XML to be transmitted with little or no modification. The simplest form is just to send an XML document from sender to receiver using HTTP which is known as *Representational State Transfer* (REST) (see section 4.1). In that case both parties have the schema information. This is not a lot different than using a binary format since the communication is useless for anybody that does not understand the format. The advantage though would be that there is no conversion from XML into another format necessary. For more advanced scenarios it becomes more feasible to wrap the document that is being transmitted into a standardized transport package or message. The most common way to achieve this for XML is by using *SOAP* which is the case in the *Transportation Security SensorNet* and described in section 4.2.

3.2 Open Geospatial Consortium

The *Open Geospatial Consortium* (OGC) is the de facto authority on open standards for *Geographical Information Systems* (GIS). Its members develop interface specifications for geographical applications. One of the primary goals is interoperability: research and development costs are later diminished by the fact that if one application implements an OGC standard other applications can use it through the predefined interfaces that the standard provides. Furthermore, there is a higher interest in the actual implementation of standards since a majority of the industry agreed upon them. This mitigates one of the main risks that proprietary applications otherwise face, the lack of

user and industry acceptance.

Some of the industry needs cover a wide area of topics whereas others are very specific. For example, there needs to be a standard for dealing with times, locations and their formats which is something that almost all geographical applications face at some point. On the other hand, the format for requesting live feeds from a sensor is of interest only to a smaller group. The OGC tries to cover everything from simple to complex that could enhance the development of spatial information applications and services.

The way it is able to achieve this is by not actually implementing the standards but only providing the framework, the specification and schemas. The usual development framework looks as follows.

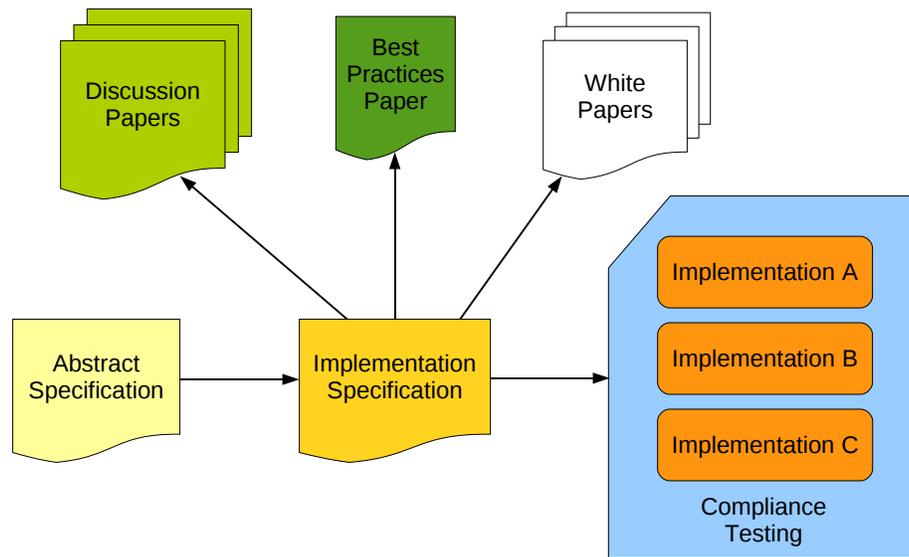


Figure 3.1. OGC standardization framework as described in [74]

First, *abstract specifications* are written that describe the goal and primary concepts of a proposed standard. This is explained in detail by Reed [74]. Second, the *abstract version* of a standard leads to an *implementation specification* which eventually becomes a *standard* after it has been accepted by the OGC members. Third, the industry

in terms of application and service developers implements the *specification* and provides feedback to the consortium. Furthermore the OGC releases *white papers* that provide high-level overviews of the concepts of a standard and a *best practices* paper that describes implementation specific development patterns. So-called *discussion papers* are usually written by developers talking about the technologies and approaches used in their implementations. Finally, the OGC encourages *implementations* to be tested and marked as compliant using their test suites.

An overview of the procedures and the approaches taken are described in the *OGC Reference Model* (ORM) by Percivall et al. [71]. It explains the concepts behind storing geospatial information, referencing locations and times, and creating maps or so-called *geometries* from the available data. The reference model refers to several abstract specifications in order to establish a connection between them and reiterate the goal of developing open interoperable standards. Apart from talking about the approaches behind geospatial information processing, the concepts of *geospatial services* and reusable patterns are introduced.

The *Transportation Security SensorNet* aims to be open and interoperable. It uses the interfaces and elements defined in the specifications of the *Open Geospatial Consortium* and provides concrete implementations, for example the *Sensor Observation Service* and the *Sensor Alert Service*. In terms of web services within a *Service Oriented Architecture* the following standards are of importance.

3.2.1 Sensor Web Enablement (SWE)

One of the main focuses of the OGC in recent years has been the development of concept called *Sensor Web*. In the *Sensor Web Enablement* (SWE) architecture and overview document by Botts et al. [10] it is described as follows:

“A Sensor Web refers to web accessible sensor networks and archived sensor data that can be discovered and accessed using standard protocols and

application program interfaces (APIs).”

This is best visualized by the concept figure 3.2 from the document.

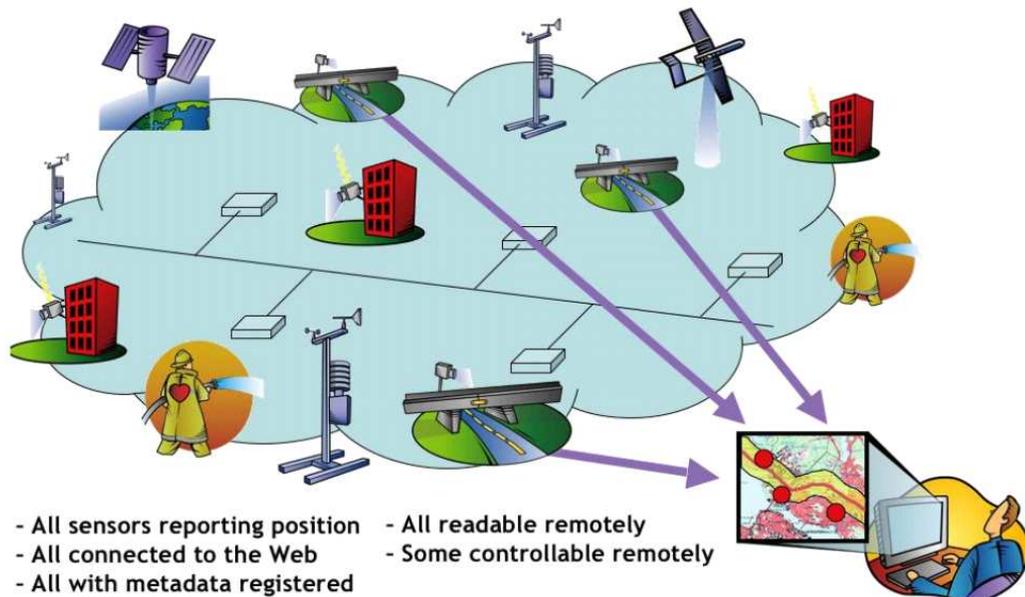


Figure 3.2. Sensor Web Concept from [10]

The idea is to combine various information modeling specifications with the appropriate services that provide the data processing for them. According to Botts et al. [10] the following specifications make up the *Sensor Web*:

- Observations & Measurements (O&M) specifies the representation of sensor measurements.
- Sensor Model Language (SensorML) describes sensors, models and their processes. For instance the discovery of sensors and data preprocessing.
- Transducer Markup Language (TML) specifies the encoding and transport of streaming sensor data in real-time scenarios.
- Sensor Observation Service (SOS) provides interfaces for describing what capabilities a sensor can perform and for retrieving actual observations or measurements.
- Sensor Planning Service (SPS) allows users to query the *sensor web* for a specific need. For example: “monitor the following 5 intersections every minute for excessive traffic for the next week”.

- Sensor Alert Service (SAS) provides users with the ability to subscribe to certain sensor events. Like “notify me when the temperature exceeds 100°F”.
- Web Notification Service (WNS) describes message exchange capabilities between clients and services.

This thesis uses the same approach in order to define the *Service Oriented Architecture for Monitoring Cargo in Motion Along Trusted Corridors* called *Transportation Security SensorNet*. However, it has to be noted that there are some differences in the implementation and use of specifications. For instance only a subset of the *Sensor Web* specifications are actually used.

The *Geography Markup Language* (GML), that is only briefly mentioned by Botts et al. [10] in the SWE document, essentially describes some of the main components and elements that are used by most of the specifications in the implementation. Additionally, the *Catalogue Service for Web* (CSW) can provide a so-called *service directory* of available services. The *Sensor Web Enablement* is an initiative from the OGC that aims at the combined growth of these specifications that will essentially make up the *Sensor Web*. While some of the specifications are agreed standards others like the *Sensor Alert Service* (SAS) are still in draft stage as of summer 2009.

Specifications that are relevant to the *Transportation Security SensorNet* are explained in more detail in the following sections.

3.2.2 Geography Markup Language (GML)

The need for a standard to encode geospatial features in an abstract way that can eventually be mapped onto real world things is elementary. The *Geography Markup Language* (GML) as described by Portele [72, 73] aims at defining most, if not all, features with a geographical background that can be defined. Among the things covered in the specification are *observation models*, *spatial and temporal reference systems*, *geometries* and *units of measure*. It considers a variety of base components that are common between applications and allows for other domain or application specific profiles to be

defined, therefore extending them. Application schemas describe a certain subset of definitions within the standard but might introduce new or extended types that are specific to the application.

The specification is highly hierarchical in the sense that several abstraction layers have been introduced in order to hide complexity. The two base objects that are defined from which all others are derived are *abstract object* and *abstract gml*. Basic types like *features* that model things like roads or rivers add more properties onto the base objects. This extension might be as simple as adding a location name and reference to it.

Things that can be modeled mathematically are part of a so-called *geometry*. This includes *points* which are *primitives*, *lines* and *curves* which are *aggregates* and can lead to more *complex* elements like *polygons* and *surfaces*.

Another big part of the specification is describing temporal constructs like *time instants*, *periods*, *intervals*, *durations* and *calendars*. Coordinate reference systems may be used differently throughout the world therefore definitions for them are included as well. They are used to specify time and location formats for instance. Units of measure are standardized definitions of measures and values of objects. There is also a section in the GML specification called observation which covers mostly simple types of observations. A more in-depth specification covering this is the *Observation & Measurements* (OM) specification (see 3.2.4).

An article by Bardet and Zand [2] gives an excellent example of how data is converted from format called AGS into GML. The main problem that is described is the lack of systematic archiving and exchange of drilling data. Since obtaining this data can be very cost intensive it has become a big issue. Hence, transforming the data into GML allows companies and researchers to take advantage of OGC applications for storage, exchange and visualization of this information. This reduces cost and makes the drilling data more useful. The article represents a case study in the sense that it describes in detail all the steps that were taken to implement the data conversion.

GML is used by many other specifications as the basis for describing geographical

information. In the *Transportation Security SensorNet* it is used by the *Sensor Observation Service* and the *Sensor Alert Service* implementations provided by, among others, the *Sensor Node* at the *Mobile Rail Network* (see section 6.3.1).

3.2.3 Catalogue Service for Web (CSW)

The *Catalogue Service for Web* (CSW) as specified by Nebert et al. [63] describes the “discovery, access, maintenance and organization of catalogues of geospatial information and related resources”. It manages resource information for services in the form of metadata.

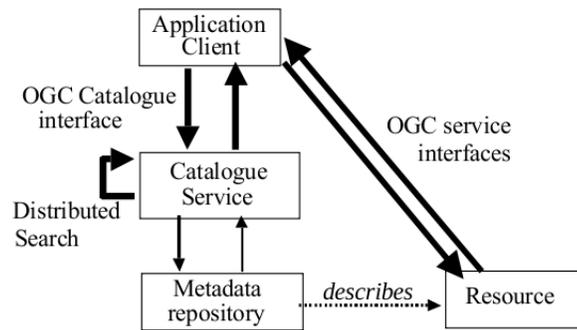


Figure 3.3. Catalogue Service reference model architecture from [63]

Whenever a client requires geospatial information or processing capabilities it queries the *Catalogue Service*. A metadata repository is kept in order to store information such as location, capabilities and schema definitions of services. Information that matches the query is then returned to the client. The client also has the ability to ask for a description of specific metadata elements and use that to get more specific results. The CSW therefore acts as broker between the clients and the services. Once the client has found a suitable service, it looks into the metadata that describes a particular service and uses that information to perform its request.

One of the advantages of this architecture is the ease of use for the client. A lot of services could provide essentially the same functionality. After they have all registered

with the *Catalogue Service* it is up to the client to choose which one to use. If a service is not available the client can simply try a different one. Furthermore it is not necessary for the client to actually know where the services are all the time since the *Catalogue Service* stores this information. This allows for a flexible environment and makes it scalable.

In the *Transportation Security SensorNet* this *service directory* functionality is provided by an implementation of *Universal Description, Discovery and Integration* (UDDI) specification (see section 4.4). Clients and web services in the framework have the option to contact it and retrieve similar information to the one offered by a *Catalogue Service*.

For additional scalability the specification also describes an approach called *distributed search*. Multiple *Catalogue Services* can set up a query topology where each service is responsible for its own metadata but the query is answered collectively. For the schema definitions of the *Catalogue Service for Web* see Nebert et al. [62].

3.2.4 Observations & Measurements (O&M)

Since there exists a variety of different sensors for almost every application, defining a standard that is true to all of them can be quite hard. The goal of the O&M standard as specified by Cox [20, 23] is to build an abstraction layer model that allows users and other services to use whatever granularity they need.

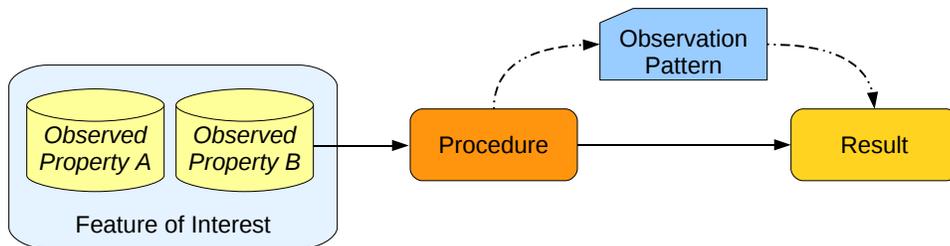


Figure 3.4. Observation process as described in [20]

Whenever an action is performed we basically “observe” a *feature of interest*. What we are interested in is the value of an *observed property* of that feature and in order

to determine this property value we exercise a particular *procedure*. Additionally an *observation pattern* can be useful for estimation and error correction of the observation result. In cases where the result is numeric the term *measurement* is used instead of *observation*. There are other specialized result types ranging from simple to complex. An observation may also be associated with a location. This is quite common.

Depending on the properties of its members, collections of observations can be one of the following types:

Type	Feature	Sampling time	Observed properties
complex	same	same	different
time series	same	different	same
discrete coverage	same	same	elements of a larger feature

Table 3.2. Collection types from [20]

The specification deals with collection types where the *feature of interest* does not change but stays the same. We speak of a *complex* observation when different properties are observed at the same time whenever a sample is taken. In case a particular property is monitored over a certain time period and the property does not change throughout the observation, the collection is called a *time series*. Sometimes the *observed property* we are interested in is made up of many smaller *observed properties*. This scenario describes a *discrete coverage*. An example given by Cox [20] is the observation of temperature values in a particular region where there are multiple sensors in the region but one is only interested in the temperature for the entire region.

Another thing described in the specification is the fact that in many cases the single *observed property* is not actually what is wanted but rather just something indirect. The *sampling of features* concept that deals with this is described by Cox [21, 22]. On the one hand, the *observed property* value could be in need of adjustment or only usable after the application of an algorithm as is often the case with light and temperature values. On the other hand, one value might not be of any importance at all but is just a part of a bigger *sample design*. Sometimes both cases can apply at the same time.

When an observation falls into this category the *sample features* form a particular relation that connects them and a so-called *survey procedure* is defined. This process achieves the desired abstraction where at a higher level the result of this relation looks like just another value since the *sampling of features* works transparently underneath it.

The *Observations & Measurements* (O&M) specification is used by the *Sensor Observation Service* in the *Sensor Node* at the *Mobile Rail Network* (see section 6.3.1). It is used in combination with GML because O&M allows for more complex observations while GML provides a broader field of geographical elements.

3.2.5 Sensor Observation Service (SOS)

The *Sensor Observation Service* (SOS) is described by Na and Priest [60, 61]. It aims to provide the user with observation data in a generic way that allows the use of a variety of different sensors. The two major types mentioned in the specification are *in-situ* and *remote* sensors. The primary goal is to provide access to observations (see 3.2.4). An implementation of this service within the *Transportation Security SensorNet* is provided by the *Sensor Node* (see section 6.3.1).

The service provides so-called *observation offerings* to users and applications. It does this by maintaining a *sensor registry* that contains information such as type, location and other metadata about the sensors that it knows about. This allows clients to perform detailed inquiries about possible observation times, available properties and geographical information of sensors and features.

GML is used to deal with measures and units in the offerings and when referencing observations. Apart from allowing filtering by sensor id the *Sensor Observation Service* is able to filter by spatial, scalar and temporal expressions. The two concepts it uses are called *data publishing* and *data consumption*.

3.2.5.1 Data Publishing

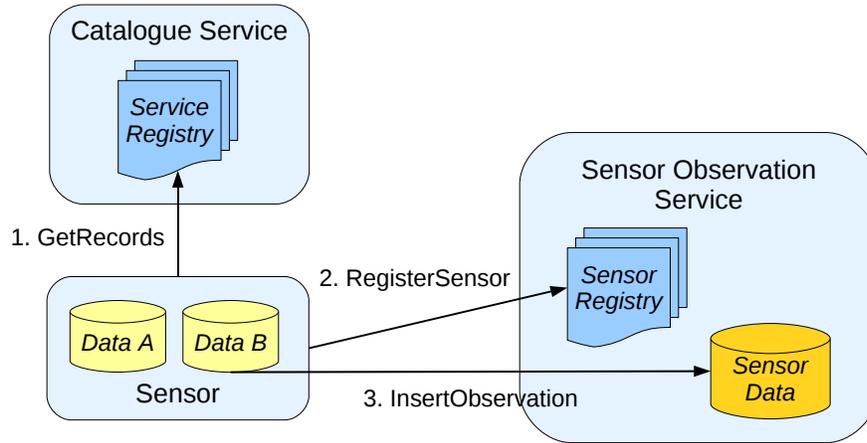


Figure 3.5. SOS data publishing process as described in [60]

The data publisher, usually a sensor, is querying the *Catalogue Service for Web* (CSW) for available *Sensor Observation Services*. After it found a suitable one it registers itself and is then able to publish data. In addition, the new sensor is automatically integrated in new *observation offerings*.

3.2.5.2 Data Consumption

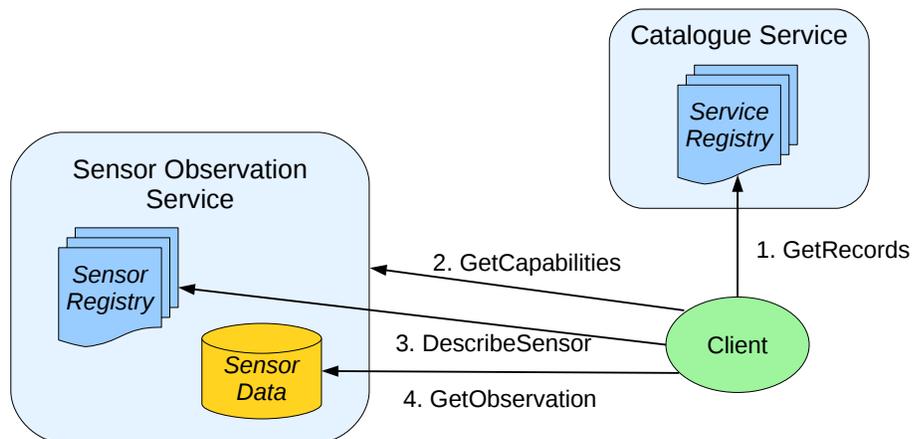


Figure 3.6. SOS data consumption process as described in [60]

The user has identified a need for a particular observation. The *Catalogue Service for Web* then provides *Sensor Observation Services*. Depending on the availability of metadata in the catalogue the user has either already selected a particular sensor or retrieves that information about a sensor from the *observation offerings*. More specific information about a particular sensor can be requested as well. Finally the necessary observations can be retrieved.

3.2.6 Sensor Alert Service (SAS)

In order to allow for an asynchronous alert reporting mechanism to notify users, the *Sensor Alert Service* (SAS) which is a candidate specification by Simonis and Echterhoff [78] has been designed. It proposes an event subscription and notification system that publishes sensor data based on specified criteria. An implementation of this service within the *Transportation Security SensorNet* is provided by the *Sensor Node* (see section 6.3.1).

3.2.6.1 Advertising Process

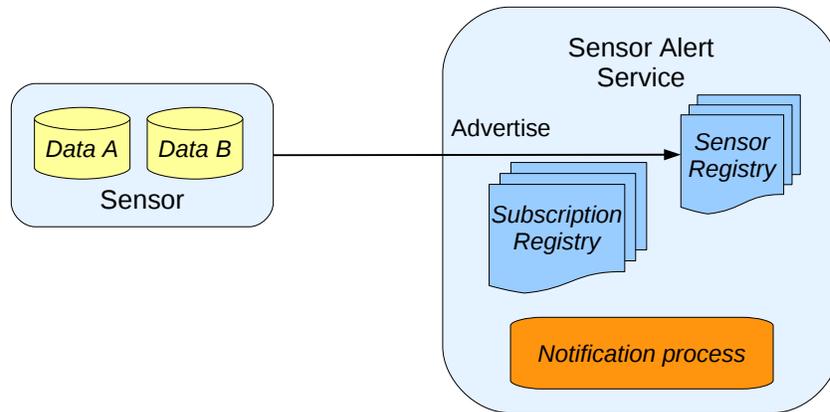


Figure 3.7. SAS advertising process described in [78]

The idea is that sensors advertise their data to the SAS. They then enter into an *advertisement* agreement to publish this data whenever it becomes available.

3.2.6.2 Notification Process

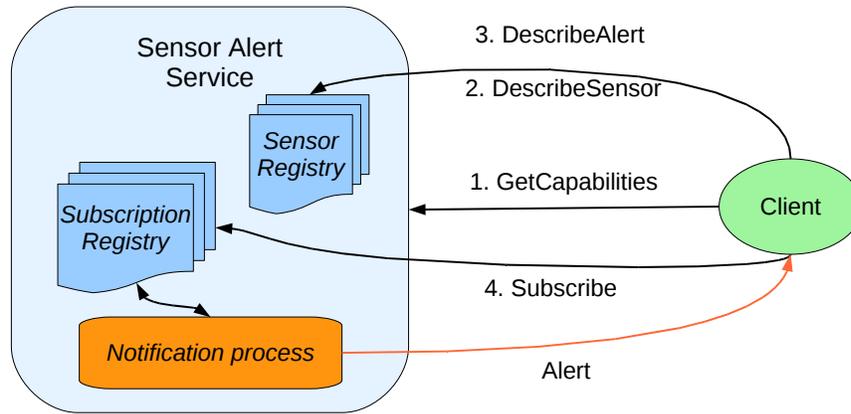


Figure 3.8. SAS notification process described in [78]

For the client, the service provides so-called *subscription offerings*. By choosing a particular offering the client subscribes to the sensor data that is defined by the offering. The SAS may modify or apply algorithms to the original sensor data which is in a way similar to applying an *observation pattern* as described in the O&M specification (see 3.2.4). The offerings are linked to *subscription criteria* that are used internally to match the sensor data that is published by the sensors to the individual clients that subscribed to them. The *Sensor Alert Service* additionally provides the client with means to retrieve all necessary information about the sensor itself and the alert data, especially the format.

The main difference between the *Sensor Observation Service* and the *Sensor Alert Service* is the way query results are provided. If the client is in need of particular sensor data on an ad hoc basis, it asks the *Catalogue Service for Web* for a matching SOS and queries the SOS in order to fulfill this need. The key aspect for the *Transportation Security SensorNet* is that the SOS only deals with providing the sensor data synchronously.

In case an alert system is needed to monitor whenever some sensor data reaches a

critical value the client does not directly act as the one querying for sensor data but rather the SAS. The client simply tells the SAS the necessary criteria for an alert through the means of a subscription. The SAS then monitors incoming sensor data and sends out notifications accordingly. This is done asynchronously without the client having to constantly query for data itself.

Chapter 4

Service Oriented Architecture

The main idea behind *Service Oriented Architecture* is that applications are defined as so-called *web services* which communicate with each other using a set of predefined protocols and standards. In terms of technologies, programming languages and platforms used, these *web services* can be completely independent systems. The key here is that their interfaces are specified using web service standards.

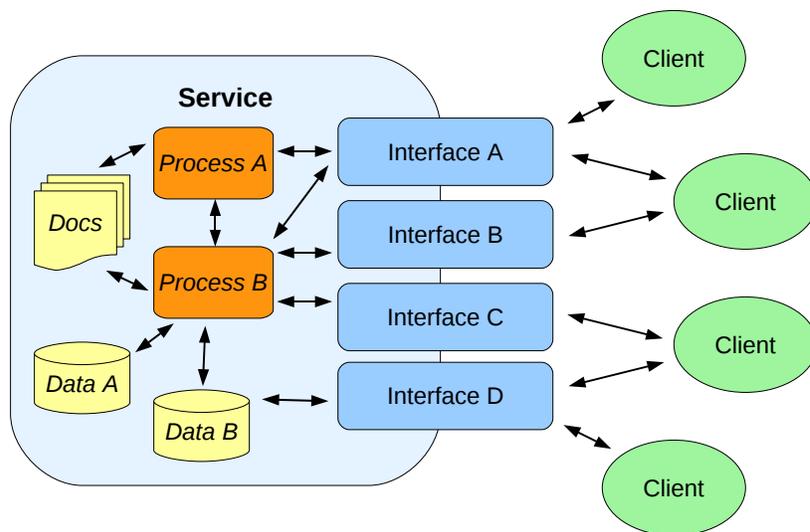


Figure 4.1. Service overview

The book “Service-Oriented Architecture: Concepts, Technology, and Design” by Erl

[26] describes these fundamentals in more detail. In particular the main components that make up a *Service Oriented Architecture* are outlined here.

A *message* represents the data that is required for a so-called *unit of work*. An *operation* covers the logic that processes these *messages*. The grouping of logic that handles related *units of work* is defined as a *service*. Additionally, the book defines a *process* as the business logic that combines several *operations* in order to complete a larger piece of work. Erl not only covers the basic concepts of SOA but also explains how they can be applied in the real world.

The principles of service orientation according to Erl [26] consist of the following:

- *Reusability* of logic, operations and services
- *Contracts* that define the service and information exchange
- *Loose coupling* of relationships with the goal of minimizing dependencies
- *Abstraction* that hides implementation logic of services
- *Composability* of services to form a more complex process
- *Autonomy* of logic within a service
- *Stateless* use of information in a service
- *Discoverability* of services

The SOA approach allows for what is called *loose coupling* between services. It defines each individual service in two ways. First, a service provides a specific functionality that could be for instance data processing or information storage. It is *autonomous* in doing so which means that it only depends on itself for providing this functionality. Second, each service can be replaced by a different service that has the same interface. This *flexibility* allows the user to choose between services based on cost, performance or availability.

Because the functionality of an entire business process or system often depends on things like cost, availability and quality of a service, so-called *service contracts* can be

defined that allow for the combination of several services into a more complex system that adheres to specific constraints. This is often necessary given the highly dynamic environments of distributed, mobile, grid and peer-to-peer systems.

The *Service Oriented Architecture* is especially useful when dealing with legacy applications. Since the entire application or system can be “hidden” behind *interfaces*, the integration or *encapsulation* of it into current business models requires far less effort. Instead of converting or rewriting a complete application, web service interfaces for it can be defined so that it becomes usable as a web service.

As mentioned before, two of the most important concepts in a *Service Oriented Architecture* are *autonomy* and *flexibility*. In addition, SOA is very cost effective because web services by default are built in a *reusable* way and because of the idea that the most *optimized* service which provides the desired capabilities is chosen. Furthermore SOA is highly scalable since it allows for the easy integration of *broker*, *proxy* and *load balancing* scenarios.

The *statelessness* principle can be seen as a rather soft requirement since there are instances of when a service needs to maintain at least some sense of state. An example would be an “online time series data processor” that looks at a specific time window in order to find patterns. It needs to keep track of the data parts that make up the window and therefore information across multiple messages.

Most of the *Service Oriented Architecture* deployments make use of at least some sort service registry that contains metadata about services and allows them to be *discovered*. The most standardized approach is the use of *Universal Description, Discovery and Integration* (UDDI) (see section 4.4) although a recent investigation by Al-Masri and Mahmoud [1] found that of all the web services that were discovered 72% can be found using web search engines and only 38% are registered in UDDI Business Registries.

Since SOA itself is a concept, several so-called *Web Services* (WS) specifications have been developed that deal with the different aspects of it. One of the most notable standards is *WS-Addressing* (see 4.3.1) which describes how routing information can be

directly attached to messages. Another one is *WS-Security* (see 4.3.3) that provides *end-to-end* message integrity and confidentiality.

The benefits of SOA according to Newcomer and Lomow [64] and their relationship to the *Transportation Security SensorNet* can be summarized as follows:

- *Efficient development* through modularity because services can be implemented independently and solely on the basis of contracts and service descriptions. This allows for tasks and implementations of clients and web services in the TSSN to be split up among team members.
- *More reuse* since it is based on open standards, *loose coupling* and platform independence. The implementation is being made available to everyone and represents an reference example as to how web services can be utilized in sensor networks.
- *Simplified maintenance* in the sense that modifications to the implementation do not necessarily change the service because of abstraction and the fact that clients utilize the service only through interfaces. With the core of the web services in the TSSN being implemented, further development can be focused on specific aspects such as security and enhancements without breaking the current system.
- *Incremental adoption* since legacy applications can be “wrapped” into a service and single applications can be transitioned into the *Service Oriented Architecture* step-by-step. This is of importance to the *Trade Data Exchange* as it needs to acquire cargo and route information from already existing systems (see section 6.5).
- *Graceful evolution* because service interaction is only interface based and services can easily be replaced by faster, cheaper or more complex implementations. With new technology and hardware becoming available parts of the current implementation of the *Transportation Security SensorNet* may be upgraded easier.

4.1 Representational State Transfer (REST)

REST is one of the major steps away from *Remote Procedure Calls* (RPC) and towards scalable and distributed web service architectures. Even though *Service Oriented Architectures* most often make use of the more flexible SOAP and its surrounding *web*

services specifications, as is the case with the *Transportation Security SensorNet*, REST still plays an important role and is widely supported.

4.1.1 Traditional Definition

The *Representational State Transfer* (REST) concept was first introduced by Fielding [30]. It originally describes the following elements:

Data Elements A *resource* represents the main data element. It can be anything like information, data or image. A *resource identifier* is used to uniquely map to a particular *resource*. In order to know what the *resource* actually is, so-called *representations* are defined.

Connectors According to REST, all interactions between a client and server are stateless. This makes it highly scalable since the server does not need to keep state information. Additionally, multiple requests at the server can be handled at the same time. Furthermore, requests can be cached, transferred by intermediaries and reused.

The original definition of request (*in*) and response (*out*) parameters is the following. *In* parameters are control data, resource identifier and an optional representation. *Out* parameters consist of response control data, optional resource metadata and optional representation.

Components The *user agent* defines the source of the request and the *origin server* is used for so-called namespace resolution of the request.

4.1.2 Current Use

The architectural style of REST has been adapted for web services and is called *RESTful*. It is closely tied to HTTP. The idea here is that *resources* are made available through *Uniform Resource Identifiers* (URI). The *representation* in most cases is XML but can also be specified using so-called *Multipurpose Internet Mail Extensions* (MIME)

types. HTTP methods such as POST, GET, PUT and DELETE are used as operations for modifying the resources.

REST can be seen as an “old” standard for web services that is still in use mainly because it is easy to use and highly flexible. It has traditionally been used in environments where the communication parties need to transmit small and “relatively” simple messages. An advantage is that the requirements on bandwidth are usually smaller when using REST compared to other approaches. With the advent of *Asynchronous JavaScript and XML* (AJAX) it has seen an abundance of new application fields. This is mainly due to the fact that AJAX uses the RESTful web service approach to provide asynchronous interaction with a web server.

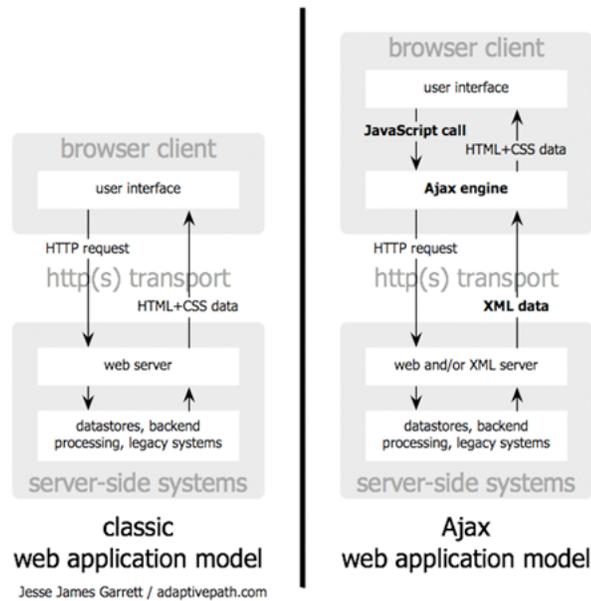


Figure 4.2. Traditional web applications and AJAX from Garrett [35]

Notable examples that use this approach are Google web applications such as *GMail*, *Maps* and *Docs*. Since AJAX is in use by entire industries, a standardization process as described by van Kesteren [83] has been started.

4.1.3 Further Development

Especially with recent developments in HTML5 as defined by Hyatt and Hickson [43] the flexibility of REST allows it to be used in more and more applications. The differences to HTML4 in terms of web application integration are significant. The enhancements described by van Kesteren [82] include *Application Programming Interfaces* (API) for playing video and audio, editing, drag and drop and more. An important addition is the ability for offline storage which allows web applications to replace desktop applications. The specification for this is defined by van Kesteren and Hickson [84]. This was currently only possible through extensions such as *Google Gears*.

All of this development and use of AJAX makes RESTful web services very appealing as they can easily be used from web applications. *Apache Axis2* which is the foundation of the *Transportation Security SensorNet* supports REST for accessing web services. This allows the use of TSSN web services in web applications without the need for additional development effort.

4.2 SOAP

The *Transportation Security SensorNet* makes use of SOAP as the default message exchange protocol. In the following SOAP is explained and a comparison with REST is made, which includes the reasons behind choosing SOAP over REST for the TSSN implementation.

According to Cabrera et al. [14] SOAP, which was formerly called *Simple Object Access Protocol*, provides “a simple and lightweight mechanism for exchanging structured and typed information between peers in a decentralized, distributed environment using XML”. It is a message standard for web services that aims to provide more flexibility and better interoperability than REST. In a comparison of SOAP to REST by Pautasso et al. [70] it was concluded “to use RESTful services for tactical, ad hoc integration over the Web (à la Mashup) and to prefer [SOAP in combination with] WS-* Web

services in professional enterprise application integration scenarios [, which is the case with the *Transportation Security SensorNet*,] with a longer lifespan and advanced QoS requirements”. The reasoning for this, including a detailed description of SOAP, follows.

One of the main differences between SOAP and REST is complexity. SOAP and the so-called *web services* (WS) specifications built around it allow for the most complex scenarios while maintaining a relatively simple basic format. REST on the other hand is usually used in point-to-point communications and the exchange of simple XML. Furthermore, one of the major drawbacks of REST is that it is tied very closely to HTTP transport whereas SOAP is not.

SOAP is independent from platforms and programming languages and allows different transport protocols to be used as so-called *bindings*. According to Nielsen et al. [67] a binding represents a “formal set of rules for carrying a SOAP message within or on top of another protocol (underlying protocol) for the purpose of exchange”. This includes describing how the protocol provides the necessary services to transport SOAP messages, how errors are handled and most importantly what *features* are provided by the underlying protocol. Although HTTP remains the most common binding, the extension of binding possibilities was one of the main enhancements to the original SOAP 1.1 specification by Box et al. [11], the other being the more clearly defined use of XML schemas.

SOAP enables extensive end-to-end message routing which is important in dealing with firewalls. The *WS-Addressing* specification (see 4.3.1) describes this in more detail. Another important aspect is security, which is available as *WS-Security* (see 4.3.3) for instance. Overall SOAP is simple in its default form yet very extensible.

4.2.1 Message format

The basic format according to the SOAP 1.2 specification by Nielsen et al. [66] defines an *Envelope* that includes a mandatory *Body* and an optional *Header* as seen in figure 4.3. The *Header* contains control information in the form of so-called *header*

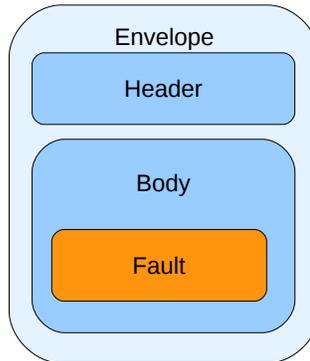


Figure 4.3. SOAP message format

blocks. These *blocks* can be used for routing or to pass processing directives to services. The *Body* is the mandatory *payload* of the message and contains the data that is being transmitted. Listing 4.1 shows the basic format that is used by all SOAP messages:

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <soapenv:Envelope
3   xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope">
4   <soapenv:Header>
5     ...
6   </soapenv:Header>
7   <soapenv:Body>
8     ...
9   </soapenv:Body>
10 </soapenv:Envelope>

```

Listing 4.1 SOAP message format example

4.2.2 Faults

Apart from the basic message format, the specification also describes the *Fault* format that is common for all messages containing error information.

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <soapenv:Envelope
3   xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope">
4   <soapenv:Header>
5     ...

```

```

6   </soapenv:Header>
7   <soapenv:Body>
8     <soapenv:Fault>
9       <soapenv:Code>
10        <soapenv:Value>soapenv:Receiver</soapenv:Value>
11      </soapenv:Code>
12      <soapenv:Reason>
13        <soapenv:Text xml:lang="en-US">Transport error: 404
14          Error</soapenv:Text>
15      </soapenv:Reason>
16      <soapenv:Detail/>
17    </soapenv:Fault>
18  </soapenv:Body>
19 </soapenv:Envelope>

```

Listing 4.2 SOAP Fault message example

The *Fault* consists of three parts. The *Code* part classifies the error into a predefined set dealing with version mismatches, so-called *mustUnderstand* header blocks, data encoding, and sender and receiver issues. The *Reason* allows the *Fault* to be described in terms of an error message and supports multiple languages. The *Details* part may contain application specific information.

4.2.3 Further development

The SOAP 1.2 Primer by Lafon and Mitra [48] includes references to several enhancements of the standard. The main reason for this is the potential for performance problems and the need for binary data transport in SOAP.

The *XML-binary Optimized Packaging* (XOP) specification by Mendelsohn et al. [58] defines the use of *MIME Multipart/Related* messages provided by Levinson [51] to avoid encoding overhead that occurs when binary data is used directly within the SOAP message. XOP extracts the binary content and uses URIs to reference it in the so-called *extended part* of the message. An abstract specification that uses this idea is the *Message Transmission Optimization Mechanism* (MTOM) by Nottingham et al. [68].

Another extension of this is *Resource Representation SOAP Header Block* (RRSHB) as described by Gudgin et al. [37] that allows for caching of data elements using so-called *Representation header blocks*. They contain resources that are referenced in the SOAP *Body* which might be hard to retrieve or simply referenced multiple times. Instead of having to reacquire them over and over again, a service may choose to use the cached objects which speeds up the overall processing time.

4.3 Web Service Specifications

The web services in the *Transportation Security SensorNet* make use of *web service specifications* in order to address topics such as addressing, event notification and security in a uniform and standardized way. The specifications that are relevant to the TSSN are described in the following sections while their implementations are addressed in chapter 6.

4.3.1 WS-Addressing

The *WS-Addressing* core specification by Gudgin et al. [39] and its SOAP binding by Gudgin et al. [38] defines how message propagation can be achieved using the SOAP message format. Usually the transport of messages is handled by the underlying transport protocol but there are several advantages of storing this transport information as part of the header in the actual SOAP message. For example, it allows the routing of messages across different protocols and management of individual flows and processes within web services.

WS-Addressing uses so-called *EndPointReferences* which are a collection of a specific address, reference parameters and associated metadata that further describe its policies and capabilities.

Addressing Header The header fields defined by the specification are the following:

- *To* which represents the destination of the message

- *From* contains the source, a so-called *EndPointReference*
- *ReplyTo* specifies that in case of a response, a message is supposed to be sent to this *EndPointReference*, which might be different from the *From* field
- *FaultTo* defines the *EndPointReference* for the fault message in the case of an error
- *Action* identifies the purpose of the message, in particular the web service operation, and is the only required field
- *MessageID* uniquely identifies every message
- *RelatesTo* references the *MessageID* of the request message in request-response message exchanges; the relationship can also be specified explicitly by defining a so-called *RelationShipType*

4.3.2 WS-Eventing

In order to allow for subscriptions to web services, the *WS-Eventing* specification has been defined by Box et al. [12]. It describes the process of establishing subscriptions as well as how the subsequent publications are delivered to the subscribers. The specification relies on *WS-Addressing* for the routing of messages. The two main components of a subscription in this specification are the *Subscribe* and the *SubscribeResponse* message. After subscriptions have been created, publications will be sent out accordingly.

Subscribe The client that wants to subscribe to a particular web service needs to define the following:

- The *Action* field of the *WS-Addressing* header is set to `http://schemas.xmlsoap.org/ws/2004/08/eventing/Subscribe`
- *ReplyTo* is the *EndPointReference* that receives the response to this subscription request
- A *MessageID* that uniquely distinguishes multiple requests from the same source
- *EndTo* defines an *EndPointReference* that is used when the subscription ends unexpectedly

- *Delivery* contains the *EndPointReferences* that are to receive the publications
- An *Expires* field that defines the expiration time of the subscription
- *Filter* that by default defines an *XPath* expression as the *Dialect*, but could be any form of expression that is applied to potential publications in order to filter them

SubscribeResponse The response to a subscription request is generated by the so-called *subscription manager*. It sends back a message with these fields:

- The *Action* field of the *WS-Addressing* header is set to `http://schemas.xmlsoap.org/ws/2004/08/eventing/SubscribeResponse`
- *RelatesTo* specifies the subscription request that this is a response to
- *SubscriptionManager* that contains its own *Address* and the unique *Identifier* for the subscription
- An *Expires* field that defines the expiration time of the subscription

The *WS-Eventing* specification also offers message constructs for the renewal, status retrieval and unsubscribing of subscriptions. Additionally a so-called *subscription end* message is automatically generated by the service that publishes information in order to notify subscribers of errors or other reasons for it being unable to continue the subscription.

It has to be noted that without additional specifications like *WS-ReliableMessaging* the delivery of publications is based purely on best effort and is not guaranteed.

4.3.3 WS-Security

The *WS-Security* specification as described by Lawrence et al. [49] deals with the many features needed to achieve so-called *end-to-end* message security. This provides security throughout message routing and overcomes the limitations of so-called *point-to-point transport layer security* such as *HTTPS*. Furthermore, the specification aims to

provide support for a variety security token formats, trust domains, signature formats and encryption technologies.

The two main aspects of security are the following:

Confidentiality This means that the information contained in a message is only *available* or *visible* to entities that are authorized. *Encryption* provides this *confidentiality* for messages.

Integrity The *integrity* of a message is maintained if it has not been modified on the way from one entity to another. Applying a *signature* enables the receiver to check if the message has been altered during the transmission.

These aspects among others are defined as part of the SOAP message. Most of the security provided by *WS-Security* is specified in header blocks of the SOAP header. The following represent its important parts:

Tokens The specification supports various types of security tokens directly:

- *User Name Tokens* for username and password pairs
- *Binary Security Tokens* which essentially are *X.509 certificates* or *Kerberos tickets*
- *XML Tokens* described by the *Security Assertion Markup Language* (SAML) or *Extensible Rights Markup Language* (XrML)
- *Encrypted Data Tokens* in which case the token itself is encrypted as well

A different way of specifying these tokens is to reference them. This is useful because at times the security tokens are specified in a different part or even completely outside of the SOAP message. The *WS-Security* specification defines the following most commonly used:

- *Security Token References* which can be used to wrap around non-standard implementations

- *Direct References* for using a *URI* as a reference point
- *Key Identifiers* that uniquely identify security tokens
- *Embedded References* which directly include tokens instead of pointing to them

Signatures In order to ensure the integrity of messages so-called *signatures* can be applied by the sender. The receiver is then able to check the validity of the message using this *signature*. Important properties that can be conveyed in the SOAP header using *WS-Security* are:

- *Signed Info* that defines the algorithms to be used for so-called *namespace transformations* and proper ordering of signature and encryption elements (for example, sign an encrypted message or encrypt a signed message)
- *Signature Value* containing the actual digital signature
- *Key Info* that defines the type of the signature used

The specification also allows for various forms of so-called *Signature Confirmations* to be sent out as responses to the initial messages. They can provide additional security in certain scenarios.

Encryption *WS-Security* provides great flexibility when it comes to the actual encryption of the message. It supports *header*, *body* as well as *individual block* encryption. The reason it is able to do this lies in the fact that it makes use of the following two constructs:

- *Reference List* that points to the *Encrypted Data* elements which, since they are completely independent of each other, enables different encryption techniques and keys to be used
- *Encrypted Key* which allows symmetric keys to be embedded in the message and is used for encrypting the SOAP *header*

Security Timestamps Most of the time, security policies need to make sure that it is possible to change previously distributed keys and force the ones that are not to be used anymore to *expire*. For this purpose *WS-Security* supports so-called *Security Timestamps* that can be attached to the message. Two fields are defined:

- *Created* describes the time when the message was serialized for transmission
- *Expires* defines the point in time when the security applied to this message is no longer considered valid

It has to be noted that *WS-Security* does not provide any methods for time synchronization which may potentially limit the effectiveness of *Security Timestamps* in certain scenarios.

A white paper by Chanliau [15] extends the definition of security to areas such as secure message delivery, metadata and trust management. It references the web service specifications that have been introduced to deal with these aspects of security in more detail.

4.4 Service Directory

Because web services by default are *loosely coupled* there has to be a way of for them to establish connectivity with each other. In general there are two different approaches for doing this. First, let a service A directly know about the presence and address of a service B that it seeks to contact. This can cause a variety of problems as all the addresses have to be managed manually which leads to scalability issues. Second, define a so-called service registry that keeps track of available services and acts as a mediator between clients and services.

The latter approach has been realized using the *Universal Description, Discovery and Integration* (UDDI) specification as described by Bellwood et al. [5] and is being used in the *Transportation Security SensorNet*. UDDI provides a XML based service registry and directory that provides the following:

- *Information* on web services and their categorizations, so-called *metadata*
- *Discovery* of web services based on specific criteria
- *Connection information* such as required security aspects, provided transports and operation parameters that describes in detail how to connect to a service
- *Alternatives* in case of a failure of one service

A paper by Bellwood [4] describes the main focus areas of version 3 of the UDDI specification:

Multi-registry Environments In order to allow for the logical separation of service registries, UDDI supports so-called *root* registries that act as parents to *affiliates*. Furthermore the replication of registries is supported. Whenever a web service publishes information to a registry it is able to either provide a key as a “suggestion” or have the registry automatically assign a new unique key to the information.

The UDDI also provides means for transferring the custody and ownership from one so-called *business entity* to another. This is an important aspect when it comes to handling cargo in the transportation industry. The *Transportation Security SensorNet* is able to provide this functionality by using an implementation of the UDDI.

Subscriptions Apart from the basic search interface that the UDDI provides, the specification describes two different subscription models:

- *Active* subscriptions check whether or not specified criteria of the previously defined subscriptions match current entries in the registry. This is done synchronously, meaning only when a request has been issued.
- *Passive* subscriptions allow for the registry to store so-called *asynchronous callbacks* for subscriptions. The registry checks against its entries on its own and independently of the initial subscriber. Whenever it finds a match it sends out a notification.

The *Transportation Security SensorNet* provides support for *active* subscriptions transparently to clients and web services . Web services automatically register with the UDDI when they are started. Clients are then able to use them by just specifying the type of service that they need. An according web service is then automatically handed to them using an underlying *active* subscription to the UDDI.

Policies The UDDI supports a complex policy abstraction model which main components are:

- *Rules* that define actions for when a set of particular conditions is met
- *Decisions* which comprise of a set of rules
- *Information access and control* that defines what kind of functionality can be provided with regard to inquiries, publications, subscriptions and others.

Policies are also used to enforce security although the specification acknowledges that only the *integrity* part of it is defined. This is partly due to the fact that the UDDI is supposed to be a public registry and lookup directory. For this particular purpose, the focus is more on the reliability of entries which can be ensured using *signatures*.

Advanced policy management that is able to restrict access to web services and even single operations as well as encrypted message exchanges are especially important when it comes to the scalability and production deployment of the *Transportation Security SensorNet*. Within the TSSN policy information as of summer 2009 is not yet in the UDDI but kept directly in the clients and web services.

4.5 Web Services Description Language (WSDL)

In order to allow services to interact and collaborate they need to share information about interfaces, operations, parameters, data elements and means of contact with each other. This has been addressed by the *Web Services Description Language* (WSDL). The most widely used and supported version is WSDL 1.1 as described by Christensen

et al. [17] but the newer version 2.0 provides a cleaner and more extensible specification.

According to Liu [54] the main improvements include the following:

- Renaming of some elements to express their intentions in more detail (*definitions* to *description*, *port type* to *interface*, *ports* to *endpoints*)
- Reorganizing the messages constructs that were previously disparate (definition is now part of *types*)
- Operations contain messages in a particular *Message Exchange Patterns*
- Introduction of more *Message Exchange Patterns*, see section 4.6
- Allows for *interface* inheritance

Overall WSDL 2.0 is a clear evolution and in many ways a lot cleaner but also far less supported than WSDL 1.1. The *Transportation Security SensorNet* uses WSDL 2.0 as it aims to provide an open framework that is extensible in the future. Figure 4.4 provides an overview of the core components of WSDL 2.0.

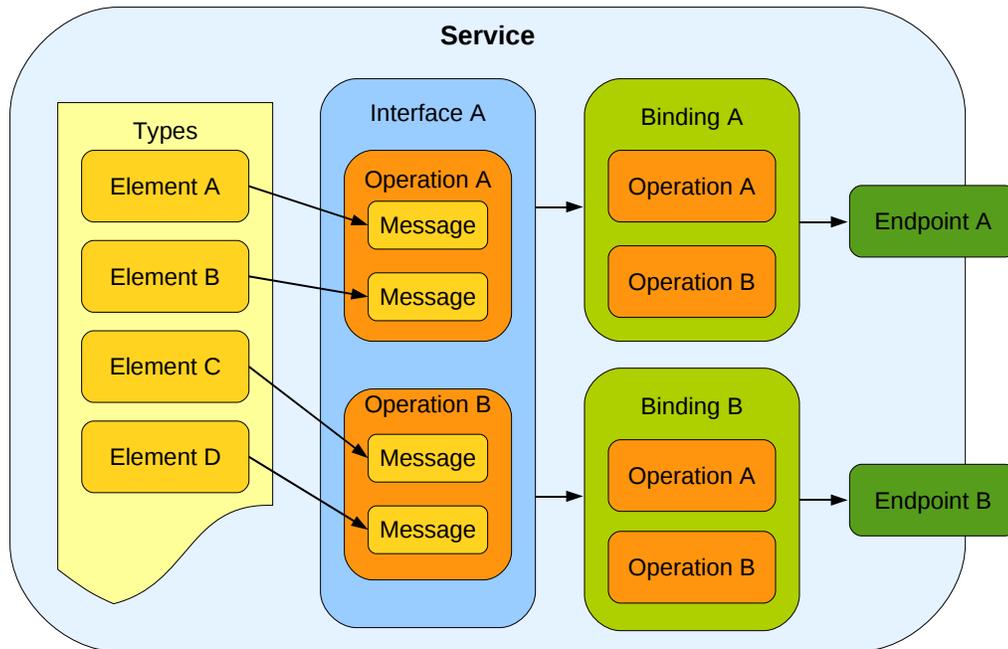


Figure 4.4. WSDL 2.0 overview

Elements that are being used by the service are defined in the *types* section. They essentially make up the *messages* of an *operation*. A group of operations then defines a so-called *interface*. A *binding* specifies the transport format for these *interfaces*. Finally the network addresses for the *bindings* are exposed as *endpoints*. Hence, a *service* can be seen as a group of *endpoints* that allow clients to use the functionality provided by the service through clearly defined *interfaces* and specified transport formats.

Interfaces from other services may be included using `<include schemaLocation="..." />` in which a location pointing to a valid WSDL file must be specified. The import namespace must be the the same as the one for the WSDL that it is included into. In order to be able to use different namespaces while still maintaining modularity, WSDL files can also be imported using `<import namespace="..." schemaLocation="..." />` and specifying a target namespace. Both of these directives are modeled after XML Schema *includes* and *imports* by Bray et al. [13].

The following is a more detailed description of the *Core Language* part of the WSDL 2.0 specification by Moreau et al. [59]. Another introduction to the main components is provided in the *Primer* by Booth and Liu [9]

4.5.1 Description

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <description
3   xmlns="http://www.w3.org/ns/wsdl"
4   xmlns:a="http://www.sample.com/elementBook"
5   xmlns:tns="http://www.sample.com/library"
6   xmlns:wsoap="http://www.w3.org/ns/wsdl/soap"
7   targetNamespace="http://www.sample.com/library">
8   ...
9 </description>
```

Listing 4.3 WSDL Description example

The *description* acts as the root for a WSDL 2.0 document that contains all other elements. It takes care of defining the target namespace and aliases for namespaces. In

the example the default namespace is set to WSDL which specifies that the document is a WSDL document. The `xmlns:soap="http://www.w3.org/ns/wsdl/soap"` references the SOAP binding for WSDL. The other namespaces that are mentioned refer to the library example which was introduced in section 3.1.1.

4.5.2 Types

```
1 <types>
2   <xsd:import
3     namespace="http://www.sample.com/elementBook"
4     schemaLocation="elementBook.xsd" />
5   <xsd:schema
6     targetNamespace="http://www.sample.com/library">
7     <xsd:element name="bookList">
8       <xsd:complexType>
9         <xs:element ref="a:book" minOccurs="0" maxOccurs="
10          unbounded" />
11       </xsd:complexType>
12     </xsd:element>
13     <xsd:element name="user" type="xsd:string">
14     <xsd:element name="error" type="xsd:string">
15   </xsd:schema>
</types>
```

Listing 4.4 WSDL Types example

XML schema elements for the service are defined in the *types* part of the WSDL. Additionally schema *includes* and *imports* are supported. The elements can then be referenced by messages later on. The code in listing 4.4 imports the *book* element from the library example which is used in the *bookList* describing a list of *books*. Additionally elements called *user* and *error* are defined in the same library namespace. Since *user*, *error*, *book* and *bookList* are fully described by the WSDL, they can now be used by both the service and the client. The service might have known about them already but by using WSDL it makes them available to clients and other services in a standardized way.

4.5.3 Interface

```
1 <interface name="LoanInterface">
2   <fault name="UserIsUnknown" element="tns:error" />
3   <operation name="getBooks" pattern="http://www.w3.org/ns/
4     wsdl/in-out">
5     <input messageLabel="Request" element="tns:user" />
6     <output messageLabel="Response" element="tns:library" />
7     <outfault ref="tns:UserIsUnknown">
8 </interface>
```

Listing 4.5 WSDL Interface example

Since version 2.0, WSDL allows for multiple *interfaces* to be defined and supports inheritance between them. An *interface* includes a group of *operations* that consist of *messages*. The *operations* must be associated with a *Message Exchange Pattern* (MEP). For more information see section 4.6. According to the MEP that is used, *input* and *output* messages are specified. They reference elements from the *types* part of the WSDL. Note that since the MEP is *In-Out* in which a *fault* would replace the response in case of an error, an *outfault* is specified. In the example an *operation* is defined that allows a user to retrieve a list of the books that were loaned.

4.5.4 Binding

```
1 <binding name="LibrarySOAPBinding"
2   interface="tns:LoanInterface"
3   type="http://www.w3.org/ns/wsdl/soap"
4   wsoap:version="1.2"
5   wsoap:protocol="http://www.w3.org/2003/05/soap/bindings/
6     HTTP/">
7   <fault ref="tns:UserIsUnknown" />
8   <operation ref="tns:getBooks"
9     wsoap:action="tns:getBooks" />
</binding>
```

Listing 4.6 WSDL Binding example

Each *binding* is able to reference the *interfaces* that were previously described in the WSDL. It associates them with a specific format and protocol that is then used to transmit messages. A *binding* can also be defined on a *operation* or even *message* level. This however is not as commonly used. The *binding* that is specified in listing 4.6 associates the *LoanInterface* with SOAP 1.2. According to the SOAP binding part of the WSDL specification by Orchard et al. [69] the *type* attribute is used to define SOAP whereas the version and the protocol (SOAP 1.2 over HTTP) are specified using the SOAP namespace. Note that for the *operation* in the example a so-called *SOAP action* is set which allows SOAP messages received by the service to be pointed to the according web service *operation*.

4.5.5 Service

```
1 <service name="LibraryService"  
2   interface="tns:LoanInterface">  
3   <wsdl2:endpoint name="LibrarySOAPEndpoint "  
4     binding="tns:LibrarySOAPBinding"  
5     address="http://www.sample.com/library/soap" />  
6 </service>
```

Listing 4.7 WSDL Service example

The last part in a WSDL document is providing an *endpoint* that specifies a network address at which the service can be reached. The same *interface* could potentially have several different *bindings*. For each of them an *endpoint* has to be defined in order to be able to use them. Hence, a service essentially exposes the defined *interfaces* and their *bindings*.

4.6 Message Exchange Patterns

In order to manage the most complex communication scenarios so-called *Message Exchange Patterns* (MEP) have been defined. They are specified for each *operation* in the WSDL document (see section 4.5.3). The basic patterns are explained in detail in

the following sections.

The *Message Exchange Patterns* are in large part based on so-called *fault propagation rules* which specify what happens in case of an error. SOAP uses them to clearly define how error messages are sent from clients to services and in between services. This allows both parties to be aware of their error handling responsibilities. The following *fault propagation rules* are defined:

Fault Replaces Message Whenever an error occurs, the message that was supposed to be sent is replaced by a fault.

Message Triggers Fault In case of an error a fault is sent back to the sender of the message. The message itself is not replaced though.

No Faults No fault is created at any time. If something goes wrong only the party that encounters the error knows about it, nobody else.

A combination of these *fault propagation rules* and the messages that are exchanged between client and service make up the *Message Exchange Patterns*. Note that whenever two services exchange messages, one is always acting as the client. Hence the MEPs depict only client-service interactions.

In the Web Services Description Language (WSDL) Version 2.0 Part 2: Adjuncts by Orchard et al. [69] the following *Message Exchange Patterns* are defined:

4.6.1 In-Only

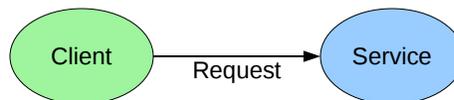


Figure 4.5. In-Only message exchange pattern

Messages in this pattern are one way only. It is defined by <http://www.w3.org/ns/wsdl/in-only>. No *Faults* are sent. This can be seen as a fire-and-forget approach.

4.6.2 Robust In-Only

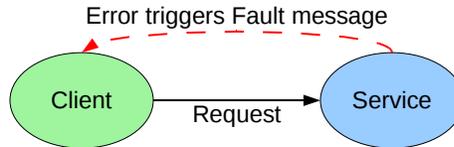


Figure 4.6. Robust In-Only message exchange pattern

This message pattern is identified by <http://www.w3.org/ns/wsdl/robust-in-only> and extends *In-Only* in the sense that it creates *Faults* when errors occur.

4.6.3 In-Out

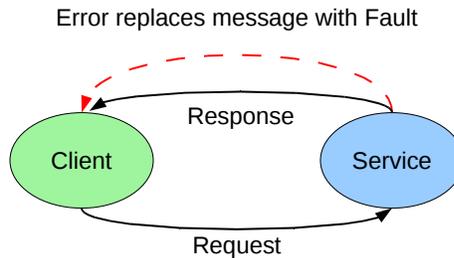


Figure 4.7. In-Out message exchange pattern

The most common *Message Exchange Pattern* is defined by <http://www.w3.org/ns/wsdl/in-out>. It specifies a request-response model where in the case of an error a *Fault* replaces the response message. Services often act as data or application providers where clients issue their requests and the service responds with either the requested data or the result of the processing that it provided.

Additional MEPs have been defined by Lewis [53]:

4.6.4 In-Optional-Out

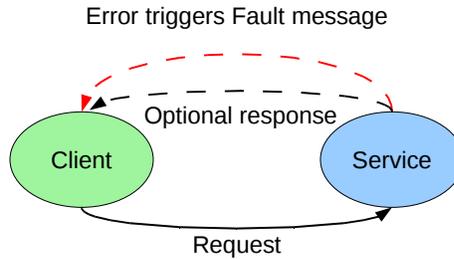


Figure 4.8. In-Optional-Out message exchange pattern

The pattern identified by <http://www.w3.org/ns/wsd1/in-opt-out> makes the response of an In-Out message exchange optional. It can be used for control messages where responses are often status messages and the assumption is that only errors are of importance in which case a *Fault* is generated.

4.6.5 Out-Only



Figure 4.9. Out-Only message exchange pattern

<http://www.w3.org/ns/wsd1/out-only> defines a *Message Exchange Pattern* that is mostly used in asynchronous communication environments and subscriptions. It is assumed that the client registered or subscribed with the service and that the service sends *notifications* back to the client at a later time. This version does not send out *Faults*.

4.6.6 Robust Out-Only

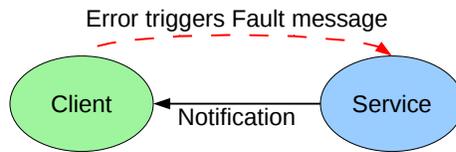


Figure 4.10. Robust Out-only message exchange pattern

In a similar fashion to *Out-Only* this pattern which is defined by <http://www.w3.org/ns/wsd1/robust-out-only> sends out messages to a client. The difference is that in case of an error it creates a *Fault*.

4.6.7 Out-In

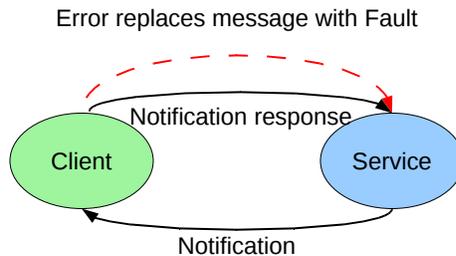


Figure 4.11. Out-In message exchange pattern

Being the reverse of the *In-Out* pattern <http://www.w3.org/ns/wsd1/out-in> describes a request-response communication that is initiated by the service. In subscription scenarios for instance the response can be seen as an acknowledgment that the notification has been received by the client. A *Fault* replaces the *notification response* in case of an error.

4.6.8 Out-Optional-In

An extension of the basic *Out-In* message exchange the <http://www.w3.org/ns/wsd1/out-opt-in> pattern provides in a sense a *selective acknowledgment* of the *notifi-*

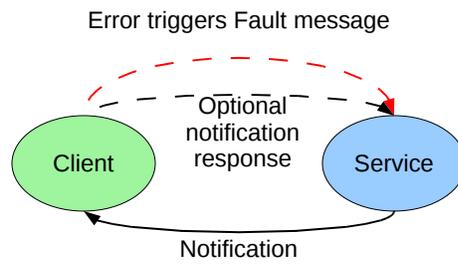


Figure 4.12. Out-Optional-In message exchange pattern

notification that was sent out. It allows for robustness by being able to send *Faults*.

Chapter 5

Related Work

In the following sections related work that is relevant to various aspects of the *Transportation Security SensorNet* such as *Service Oriented Architecture*, web services, communication models, the *Open Geospatial Consortium* specifications and sensor networks is analyzed.

5.1 Microsoft - An Introduction to Web Service Architecture

The paper by Cabrera et al. [14] about web service architectures gives an excellent introduction to what eventually evolved into the *Service Oriented Architecture*. The key ideas described are the following:

Message only approach The only thing that is exchanged between services are messages. This principle avoids potential problems that could occur when functionality embedded in different components becomes too intertwined. It also ensures flexibility and interoperability between services. The services and messages are defined in *Web Service Description Language* (WSDL) and then transported using SOAP. How the messages are sent from one service to the other is specified is so-called *Message Exchange Patterns* (MEP). Additional properties like security or reliability are standardized in the *Web Service* (WS) specifications.

Flexible protocol stack In order to provide support for a variety of systems, SOA needs a protocol layering model that ranges from general purpose to highly specific. The modular architecture of SOAP describes a protocol that consists of “building blocks”. This ensures two things. First, you only pay for what you actually use and second, it can be complemented or extended at any time.

Autonomy of services As described before, services aim to embed their functionality and be independent from each other. The extensibility of SOAP allows for the so-called *evolution* of a web service, also known as versioning. The *mustUnderstand* annotation can be provided to signal that the recipient of a message needs to know how to handle the SOAP header specifics. In order to maintain this autonomy and at the same time allow complex business models to be used, services must form *trust relationships* with the services that they use. The reason for this is that essentially there is no apparent difference between two services that provide the same interface. Businesses must know that they can trust their data to be handled confidentially by the service that they choose. Without this trust paradigm there are many potential security concerns. Another point mentioned is the move from a centralized system to a more federated approach using SOA which is able to deal better with the entire message exchange model.

Managed transparency In order to be flexible enough to support different programming languages and platforms, *Service Oriented Architectures* use a service abstraction layer model. The implementation and internal processes of a service are completely hidden from its client. The only thing visible are the so-called *interfaces* that are provided. Every service in SOA is described using the *Web Service Description Language* (WSDL). The WSDL file of a service defines its capabilities and provides a standard for the interoperability of clients and services.

Protocol-based integration The interaction between services should be restricted to the communication using a predefined protocol only. This allows for applications to be self-contained and independent of their implementation language and system. As described before it provides this by using abstraction layering through interfaces and the use of metadata. The *Service Oriented Architecture* follows the “nothing is shared” approach. This *autonomy* is the reason why it can provide the aforementioned *flexibility*.

Cabrera et al. [14] outline concepts that led to the implementation of *Service Oriented Architectures* and development of the *web services* specifications that surround them and are used by the TSSN. A lot of the main approaches have been standardized in various committees and organizations by now but were only in the early stages when this paper first came out.

5.2 Adobe - Service Oriented Architecture

An Adobe technical paper by Nickul et al. [65] outlines general architecture approaches that can be taken when transitioning business processes to the *Service Oriented Architecture*. It mentions a widely used technology called the *Enterprise Service Bus* (ESB) that provides a standardized means of communication for all services that connect to it. For the *Transportation Security SensorNet* this is of importance when it comes to asynchronous communication as the *Java Message Service* (JMS) uses queues that are on the ESB for message exchanges (see section 6.1.6 and 8.2).

In the example that is provided, three business processes all have some sort of login, authentication, name and address management. The problem that occurs most often in scenarios like this is how to synchronize states across all three processes. Using SOA this common task is bundled into a service that all three processes connected to the ESB can use which improves efficiency and greatly decreases required maintenance.

In addition to the basic *Request-Response*, several other *message exchange patterns* that go beyond the standardized ones (see section 4.6) are described:

5.2.1 Request-Response via Service Registry (or Directory)

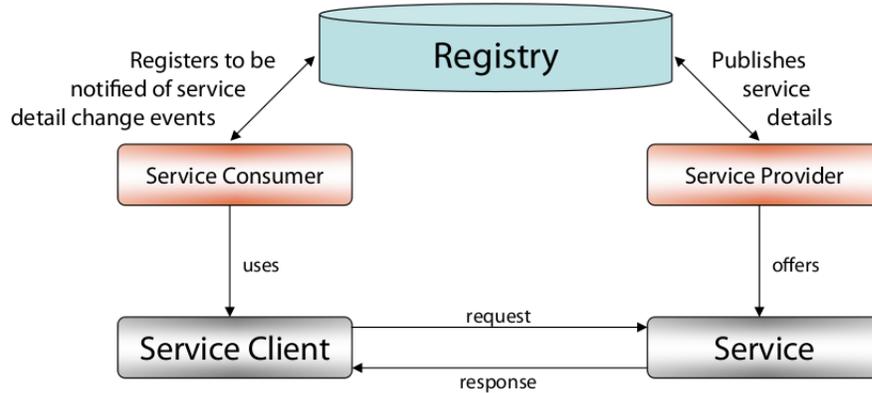


Figure 5.1. Request-Response via Service Registry (or Directory) message exchange pattern from [65]

A so-called *registry* keeps track of service metadata. The *service provider* is responsible for updating it whenever a change occurs and the *service consumer* subscribes to the *registry* for any of these changes. The metadata that is provided is then used to configure a *service client*. Hence, the client can issue *requests* and receive *responses*.

The *Transportation Security SensorNet* essentially uses a very similar approach with the UDDI. Web services automatically register with the UDDI when they are started and clients are able to use specific services by looking them up in the UDDI.

5.2.2 Subscribe-Push

The *service consumer* uses the client to subscribe to specific *events* as shown in figure 5.2. Whenever the service encounters one of these *events* it pushes *notifications* back to the client or other endpoints that were defined in the subscription. This approach is conceptually similar to what is described by the *WS-Eventing* specification (see 4.3.2).

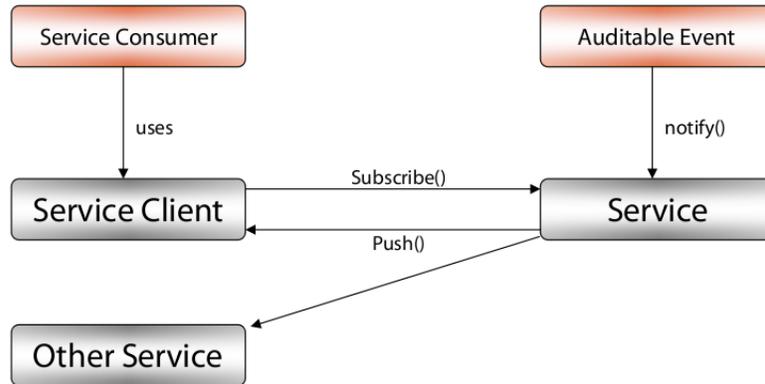


Figure 5.2. Subscribe-Push message exchange pattern from [65]

5.2.3 Probe and Match

When there is no *service registry* available, a client has to discover usable services on its own. By using multicast or broadcast messages it *probes* until suitable services respond with a *match*. A hybrid approach could use the registry for a candidate set of services to *probe*. This pattern does not scale very well because it is highly dependent on the available bandwidth.

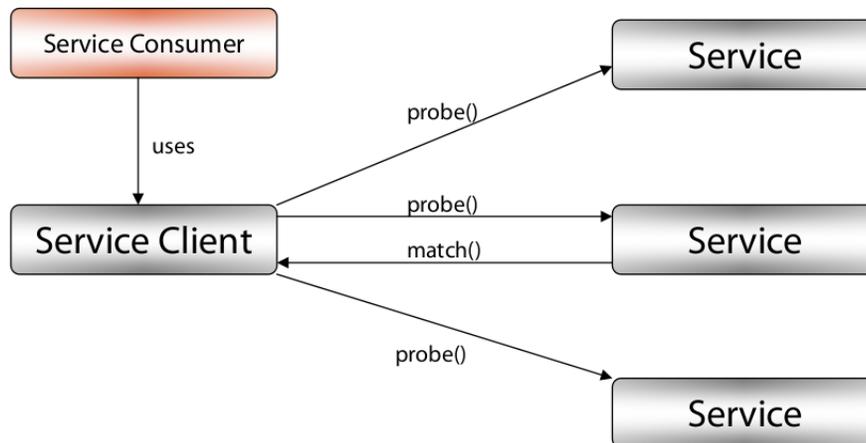


Figure 5.3. Probe and Match message exchange pattern from [65]

5.3 Open Sensor Web Architecture

An approach to implement the proposed standards of the *Sensor Web Enablement* that are described in section 3.2.1 is outlined by Chu et al. [19]. A more detailed definition of the system and its core services is provided in the thesis by Chu [18]. The system is called *NICTA Open Sensor Web Architecture* (NOSA) and is focusing on the combination of sensor networks and distributed computing technologies. For this purpose the following four layer model is defined:

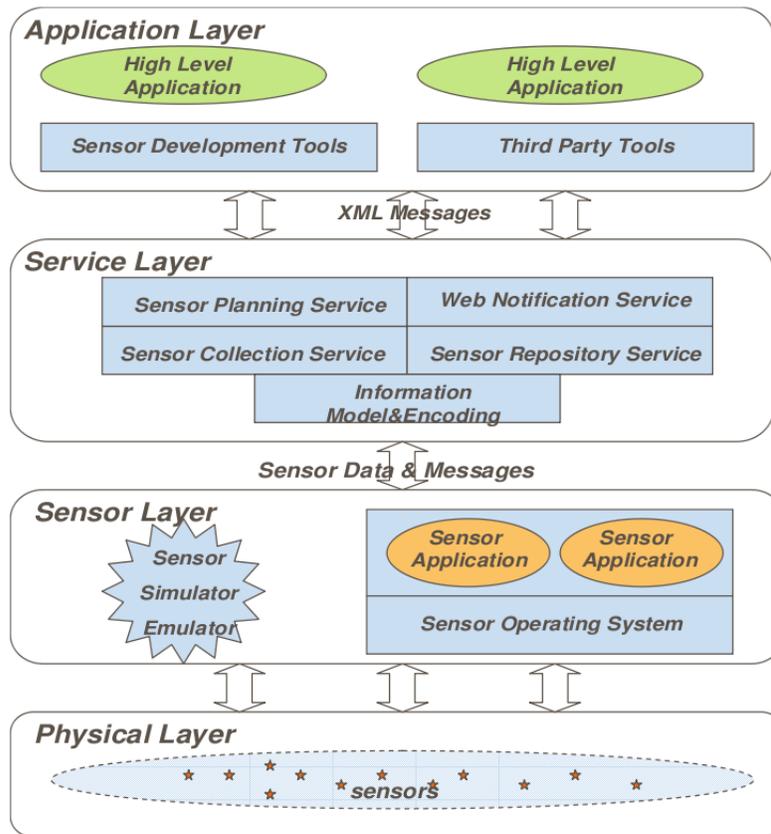


Figure 5.4. NOSA layer overview from [19]

Physical layer The sensors can be contacted using standardized means such as ZigBee and other IEEE 802.15 protocols. They can also interact with each other.

Sensor layer This layer provides the main sensor applications that are built on top of the *Sensor Operating System*. This operating system is called *TinyOS* (see Levis et al. [52]) and is widely used in low power sensor environments. It deals with the control, monitoring and retrieving of data from the sensors in the *physical layer*. The sensor layer acts as the basis for services that make use of this data.

Service layer Web services that are compliant to the ones defined in the *Sensor Web Enablement* are part of this layer. They provide a uniform and standardized way of dealing with sensors and the data that they gather.

Application layer Applications that want to interact with the underlying service infrastructure are provided with development and third party tools that to make use of the open standards web service interfaces.

The *Transportation Security SensorNet* uses a similar approach but has some significant differences. The goal of both implementations is to integrate a sensor network into a web services architecture using open standards. NOSA uses a sensor application that is tightly integrated into the *Sensor Operating System* and then provides sensor data and control to web services in a non-standard format. TSSN on the other hand implements sensor management and monitoring functionality inside a single service, the *Sensor Node* (see section 6.3.1) and allows different sensors to be “plugged in”. This allows other services to use standard web service interfaces and SOAP messages in order to access sensors.

Furthermore, the web services used by NOSA are implemented manually according to the *Open Geospatial Consortium* specifications which causes them to be limited as not everything that is specified is also implemented. In contrast, the TSSN uses automatic code generation (see section 6.1.1.4) that enables it to use all OGC specifications. Since their elements and interfaces are generated the only thing that has to be implemented is functionality. This approach significantly reduces development efforts.

5.4 Globus - Open Grid Services Architecture

Globus is an architecture that is based on grid computing. It focuses on providing capabilities as services in a grid environment using standard interfaces and protocols. An initial paper by Foster et al. [32] gives an overview of the architecture and design decisions. In particular, Globus supports “local and remote transparency with respect to service location and invocation” and “protocol negotiation for network flows across organizational boundaries”. Its service approach is similar to the *Service Oriented Architecture* that is used by the *Transportation Security SensorNet*. Additionally, security concepts that work inside a grid are applicable to SOA and vice versa.

Services Functionality in the Globus defined architecture can be achieved using so-called *grid services* which utilize standard interfaces in order to provide the following:

- *Discovery* of capabilities and the services using standardized *naming* conventions
- *Lifetime management* which includes *dynamic service instance creation* and *concurrency control* of data and processes
- *Notification* of clients and subscribers in case of events
- *Manageability* of service relationships and maintenance
- *Upgradability* in terms of versioning to ensure compatibility between services
- *Authorization* to enforce access control

Protocols The two important aspects regarding protocols that Globus deals with are:

- *Reliable service invocation* ensures that the exchange of messages which is the core of service interaction is reliable. This allows for the means of communication necessary in a grid computing environment.
- *Authentication* addresses the need to verify the identity of clients and services in the grid

The current architecture of Globus as shown in figure 5.5 is still based on the same principles that were initially described by Foster et al. [32]. The combination of custom components and web services components provides an architecture for security, data management, execution management, information services and a common runtime in a grid environment. In the following, the approaches taken are described in detail.

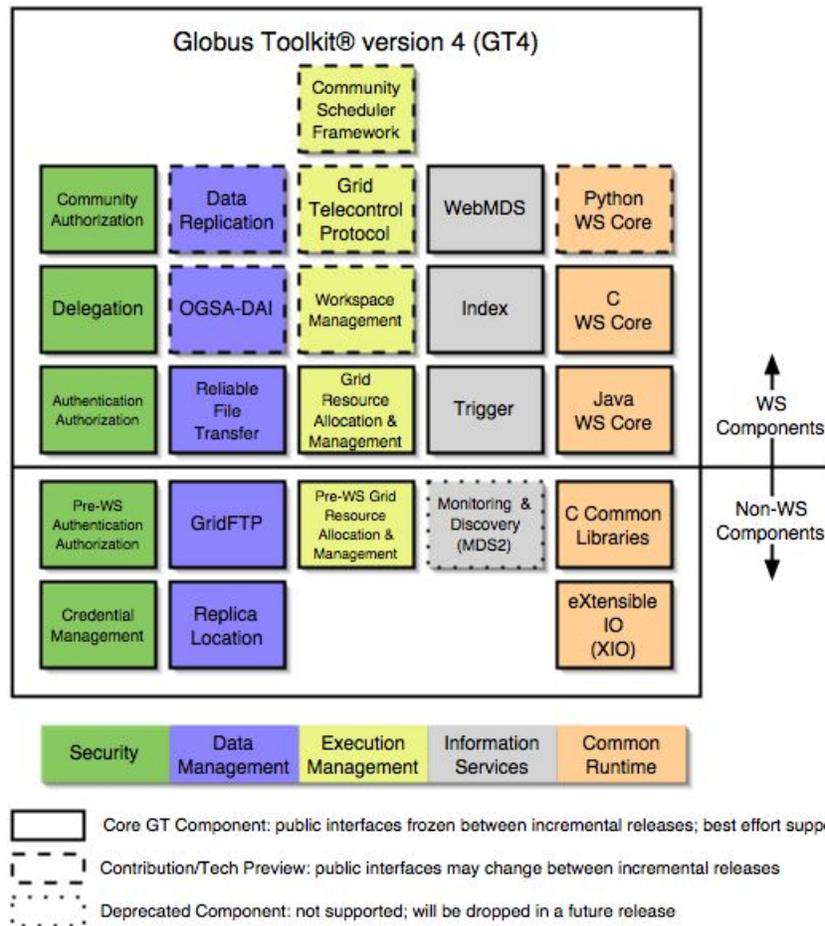


Figure 5.5. Globus Toolkit overview from <http://www.globus.org/toolkit/about.html>

Service model All entities are represented as services that provide standard interfaces over which their capabilities are accessible. Invocation of a particular functionality and the interaction between services is performed using message exchanges. These

grid services utilize *web services* specifications for their interfaces and implementations. Since a service in Globus is both, dynamic and stateful, it is assigned a so-called *grid service handle* (GSH) to uniquely identify it. In order to support the *upgradability* concept, a particular version of the service is identified by a *grid service reference* (GSR).

Factories Services in the grid that are able to create new service instances are called *factories*. Whenever a new service is created, it is automatically assigned a new *grid service handle*.

Service lifetime management Globus allows task specific services to be instantiated. These so-called *transient* services perform a predefined task and terminate upon its completion. It is also possible to associate a particular lifetime with a service. Note that services that need more time in order to complete their task may request a *lifetime extension*. An important aspect regarding the lifetime management is time synchronization across all services. In order to achieve this, Globus uses the *Network Time Protocol* (NTP).

Handles and references A so-called *HandleMap* is used to map *grid service handles* to specific *grid service references*. This is necessary since *grid service references* have a defined lifetime and may expire. The *HandleMap* ensures that it only returns valid *grid service references* and not ones that are already terminated. This among other things also allows detailed access control all the way down to the operation level. For this to work, every service needs to register with a so-called *home HandleMap*. The *grid service handle* is constructed in a way that it automatically references this *home HandleMap* to ensure scalability.

Service data and service discovery Every *grid service* is associated with so-called *service data* which in Globus is a collection of XML documents that describe the capabilities of the service. By default each service provides this data using the mandatory

FindServiceData interface. The overall system contains a *registry* that contains references to each individual service. It provides a *Registry interface* that is used to register *grid service handles*. Since the availability of services can change, the *registry* has to adapt. In order to deal with these dynamics in the grid environment, registrations must be refreshed otherwise they expire after a specified time.

Notification Globus provides an asynchronous notification system that is based on subscriptions. A client acts as a so-called *NotificationSink* that issues a request for particular events to the so-called *NotificationSource*. In the case of events, notifications are then pushed from the *source* to the *sink*.

Change management *Web services* interfaces in the grid environment are uniquely named in order to provide manageability. Whenever a significant portion of the interface or implementation is changed, a new unique name must be provided.

In contrast to the *Transportation Security SensorNet*, Globus makes use of web service specifications in some of its components but also provides custom implementations and interfaces as for service discovery and notifications. The TSSN uses web services specifications and *Open Geospatial Consortium* standards almost exclusively which ensures standards compliance and compatibility. For service discovery the UDDI (see section 4.4) is used and for notifications *WS-Eventing* (see section 4.3.2).

5.5 Service Architectures for Distributed Geoprocessing

A research article by Friis-Christensen et al. [34] deals with the integration of *Open Geospatial Consortium* specifications. It outlines the implementation of an application that analyzes the impact of forest fires using web services. The purpose of the application is to assess the damage inflicted by fires based on land cover data for a particular area. The previous solution looked like figure 5.6.

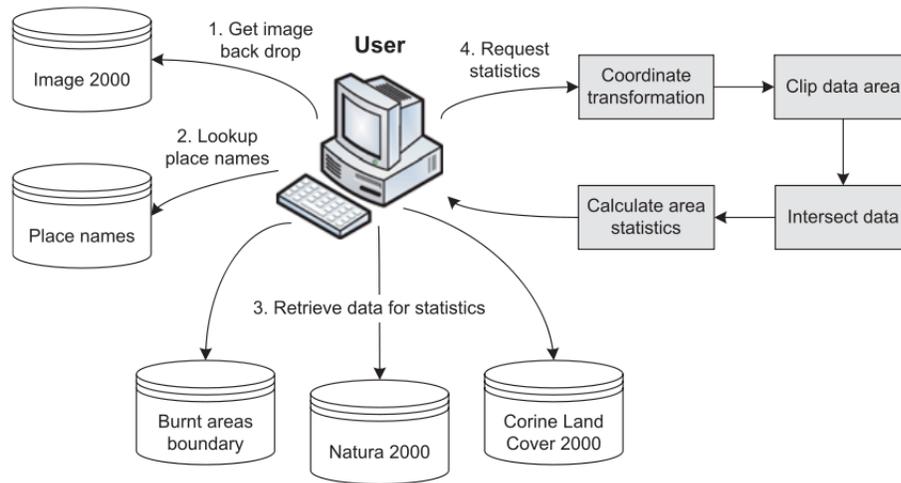


Figure 5.6. Forest fire application from [34]

Friis-Christensen et al. [34] discuss advantages and disadvantages of their improved, web services based implementation and outline potential solutions for problems that they discovered.

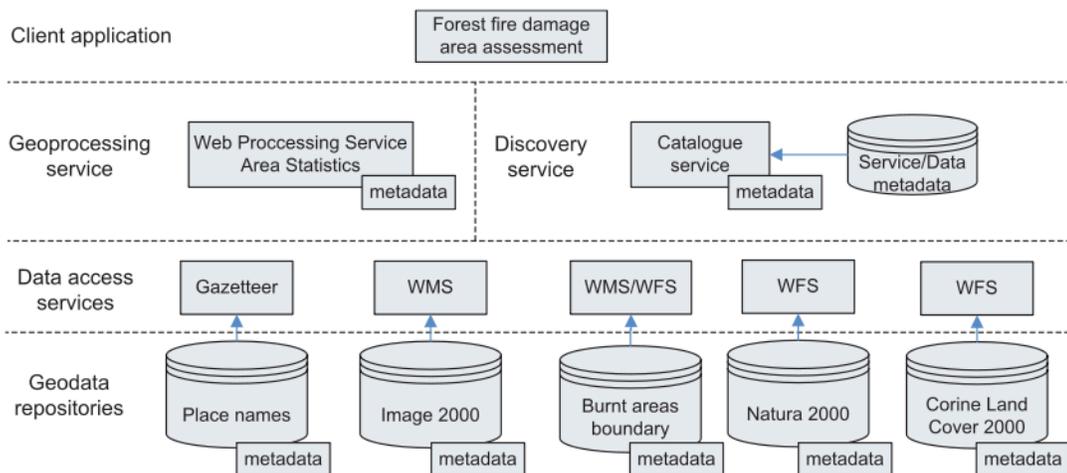


Figure 5.7. Forest fire web services architecture from [34]

Architecture The main focus is the transition from a client application to a flexible web services architecture using *Open Geospatial Consortium* specifications. As shown

in figure 5.7 the components include multiple data sources that are made available through data access services like the *Web Map Service* and the *Web Feature Service*. A geoprocessing service performs the analysis of the data and provides it to a client. Furthermore a discovery service serves as the registry for all services and their metadata.

The general process is described as follows:

1. Retrieve a map
2. Select a time and area of interest
3. Search for data source *masks* that deal with burnt areas
4. Search for target data *masks* that serve as the basis for the assessment of fire damage
5. Execute the process which retrieves the masked features, performs calculations and returns the desired statistics
6. Display statistics

Statistics Service This is the implementation of a *Web Processing Service* (WPS) according to the OGC specifications. Apart from the general *getCapabilities* interface, a *describeProcess* interface is defined which is used to explain how data is handled within a particular process and what functionality the process provides. The *execute* operation is used to start the specified process with previously defined filters, so-called *masks*, as the parameters. During the processing, the statistics service uses these masks to collect *features* from the data sources.

Mapping and Feature Services These services provide the relevant data such as satellite imagery and statistics either in its entirety or through the application of specified *masks*.

Catalogue The catalogue serves as a service registry and allows searching for services and features based on *title*, *bounding box* and *time of interest*.

Client In the implementation that is described in the paper, the client application is browser based. It uses a combination of client (AJAX) and server (JSP) based technology to display maps and the calculated fire damage statistics

The prototype implemented uses synchronous communication in between services. The problem in this case is that the actual processing can take quite a long time. In the future the authors want to transition to an asynchronous communication model that is similar to the *OGC Web Notification Service*.

In addition, it is pointed out that even though standardized interfaces allow for a combination of services which provides flexibility, the transport of high volumes of data is often not feasible in geoprocessing scenarios which can lead to highly specialized but not very reusable services.

The implementation described by Friis-Christensen et al. [34] is interesting in the sense that it exclusively uses specifications from the *Open Geospatial Consortium* which makes it compatible to other *Geographical Information Systems*. The *Transportation Security SensorNet* aims to be OGC compliant as well but includes specifications that deal with sensor networks such as the *Sensor Observation Service* and the *Sensor Alert Service*, something that this forest fire web service architecture does not even address.

5.6 Web Services Orchestration

A paper that specifically deals with the problem of reusability of services and so-called “next generation challenges” was written by Kiehle et al. [47]. The idea here is to increase transparency and reusability by splitting processes into smaller more reusable processes and utilizing a work flow management system called *Web Services Orchestration*. This is especially important for the integration of the *Transportation Security SensorNet* into systems used in the transportation industry. Its modular design and architecture allow single components to be reused and information flows to be created.

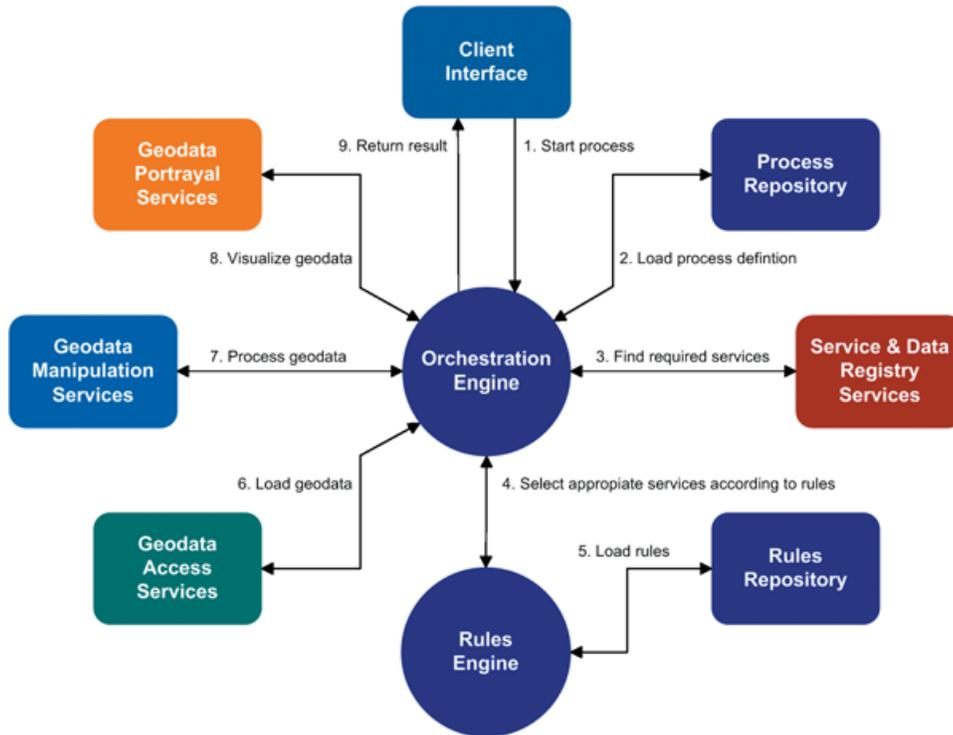


Figure 5.8. Web orchestration framework from [47]

The *Web Processing Service* specification describes how services can be arranged and combined into so-called *service chains* that form a process. Two alternatives are commonly used in order to achieve this. A *Web Processing Service* can be setup to combine and “encapsulate” other individual web services and therefore provide the desired abstraction. However, the best way to define work flows is using the so-called *Business Process Execution Language* (BPEL). BPEL enables complex *service chains* as shown in figure 5.8 to be defined without the need for custom and potentially not reusable *Web Processing Services* that just “encapsulate” services.

5.7 Summary

The related work addresses the following key technologies that play an important part in the *Transportation Security SensorNet*:

Service Oriented Architecture The development of the *Service Oriented Architecture* and its web services specifications has come a long way but is still far from over. Even though specifications exist, organizations and businesses often implement components that are similar to the specification but not compliant. As discussed before, this is the case for service discovery and notifications in Globus. Two common reasons behind this are the following. First, the specification may be available but there are hardly any reference implementations that can be used. Second, extensions to the specification that are necessary for a particular implementation or in a specific environment such as the grid are not covered by the standard.

Open Geospatial Consortium The specifications by the *Open Geospatial Consortium* are often complex and there is significant development effort necessary to implement the elements, interfaces and functionality they define. Automatic code generation as described section 6.1.1.4 and used by the *Transportation Security SensorNet* can facilitate their implementations but is not used very often.

Sensor Networks The implications on communication models that sensor networks have, in particular asynchronous message exchanges, are often ignored in web service architectures. As seen in NOSA, the focus is on the implementation of a subset of OGC standards for a particular sensor network, but the link to an overall *Service Oriented Architecture* seems to be missing.

It is evident that current systems seem to lack the combination of SOA, OGC specifications and sensor networks. The *Transportation Security SensorNet* combines all these technologies and bridges the gap between implementations that just deal with SOA and OGC specifications and systems that use OGC standards in sensor networks.

Chapter 6

Design & Architecture

6.1 Overview

This chapter describes the architecture of the *Transportation Security SensorNet* (TSSN). It provides an in-depth discussion of design aspects and the implementation.

6.1.1 Service Oriented Architecture

“Service Oriented Architecture (SOA) is a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains.” MacKenzie et al. [55]

Building a “Service Oriented Architecture for Monitoring Cargo in Motion Along Trusted Corridors” makes sense. According to a study by the Delphi Group [36], companies that collaborate usually request compliance for the following standards: XML 74%, J2EE (Java) 44% and SOAP 35%. The architecture used for the implementation of the *Transportation Security SensorNet* utilizes all three technologies by separating functionality into *web services*. This allows for high flexibility and is very cost effective (see chapter 4).

Haas et al. [40] early on proposed various *models* for web service architectures. The *Message Oriented Model* focuses on message relations and how they are processed. An

approach that centers around resources and ownership is the so-called *Resource Oriented Model*. The *Policy Oriented Model* defines constraints and focuses on security and quality of service. Ideas from all these models have been combined with the *Service Oriented Model* into what has become the *Service Oriented Architecture*. Of the proposed models it has been the most widely implemented.

A book that provides an excellent overview of Java and *web services* is written by Kalin [45]. Note that the *Service Oriented Architecture* by definition is programming language and platform independent. It is built on the basis of requests and responses and the independence of so-called *web services*. The choice to use Java for the implementation was made because the *Transportation Security SensorNet* is built on top of previous research on the *Ambient Computing Environment for SOA* by Searl [76] which is written in Java.

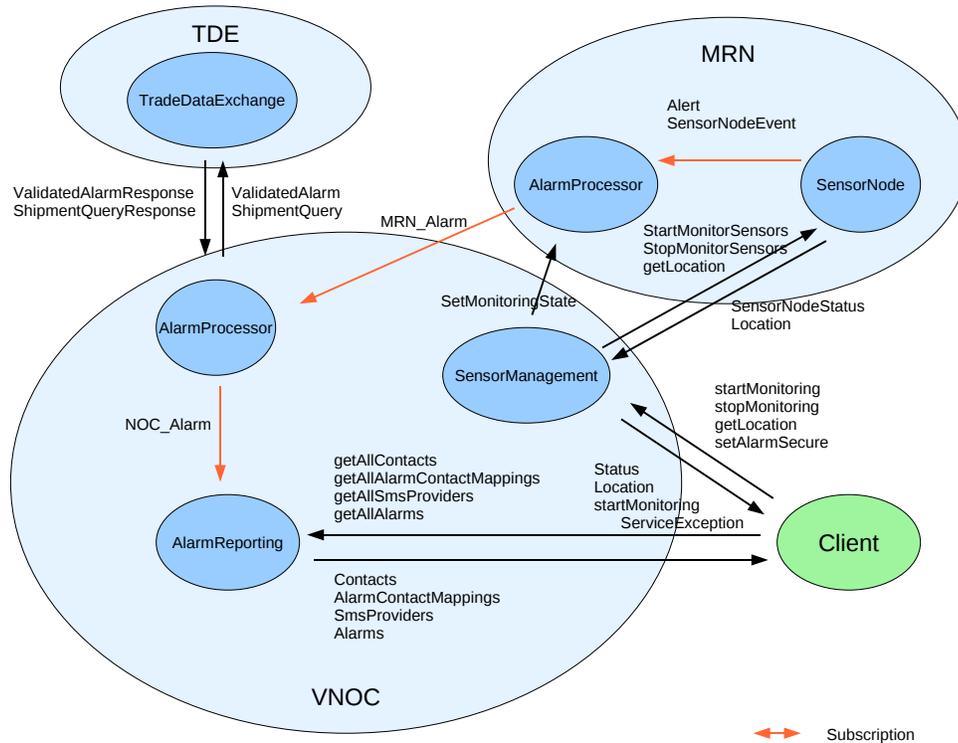


Figure 6.1. Service message overview

The main components of the *Transportation Security SensorNet* are sensor management and alarm notifications. An overview of the services and relevant message exchanges is shown in figure 6.1.

The so-called *Trade Data Exchange* (TDE) (see section 6.5) provides shipment, route, logistics and relevant cargo information. It is managed externally and used by the system only through its specified interface. The *Virtual Network Operation Center* (VNOC) (see section 6.4) is responsible for the processing of sensor data and alarms. One of the major capabilities that it provides is alarm notification. The *Mobile Rail Network* (MRN) (see section 6.3) deals with the actual management of sensors. *Web services* at the *Mobile Rail Network* capture sensor data from the sensors and “preprocess” that data. A detailed description of each individual service is provided later in this chapter.

The architecture consists of web services that are separated into so-called *service clouds*. These *service clouds* represent the different geographically distributed locations (e.g. Overland Park, KS; Lawrence, KS and on a moving train) where services are deployed and are shown in figure 6.2.

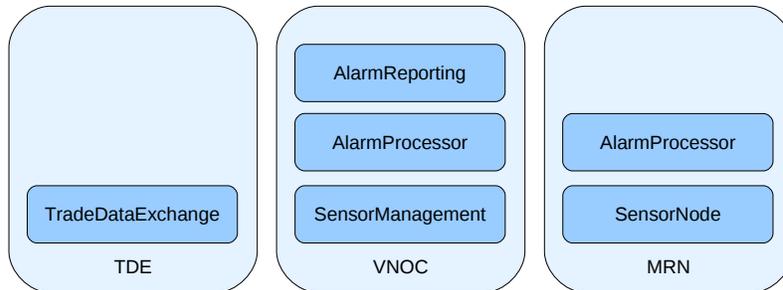


Figure 6.2. Service cloud

The *web services* are developed according to the *web service* specifications and the standards provided by the *Open Geospatial Consortium*. This means that they aim to be standards compliant. Since the OGC specifications are at times very complex, the *Geography Markup Language* for example defines over 1000 elements, the basis for

the framework was implemented using custom interface definitions first and adding the OGC ones later. This enabled fast prototyping and testing of the system.

An analysis of geospatial problems and their potential solutions is done by de Smith et al. [24]. Among other things it is pointed out that using standards, in particular the specifications provided by the *Open Geospatial Consortium*, greatly increases interoperability and allows for the development of distributed systems that are more flexible than commonly used *Geographic Information Systems*.

The following sections explain in-depth the approaches and technologies used in the implementation of the *Transportation Security SensorNet* that represents a “Service Oriented Architecture for Monitoring Cargo in Motion Along Trusted Corridors”.

6.1.1.1 Ambient Computing Environment for SOA

The infrastructure described by Searl [76] called *Ambient Computing Environment for SOA* forms the basis of the implementation of the *Transportation Security SensorNet*. It provides a complete SOAP stack using *Apache Axis2* and a variety of other useful programs that assist in the development of a *Service Oriented Architecture*.

The *Ambient Computing Environment for SOA* [76] deals with multiple ownerships and federations that provide *web services*. In particular it covers the following aspects:

- *Service Discovery* across different federations
- *Authentication* of clients and services
- *Authorization* of clients and services
- *Subscriptions*

The implementation of the capabilities provided is based on *Apache Axis2* and the *web service* specifications. It is explained in detail in the following sections.

6.1.1.2 Apache Axis2

Apache Axis2 is a software stack that allows the development and running of *web services* and clients. Its architecture as described by Chinthaka [16] consists of the following main components:

AXIs Object Model (AXIOM) AXIOM is an XML object model that aims for high performance while requiring low amounts of memory. The idea behind it is the application of a so-called *pull parser*. This allows objects to be built from XML only up to the information that is needed by the user while the rest of it is *deferred*.

The advantage of this is that the memory that an object requires is significantly reduced. Furthermore, since the entire object model does not have to be constructed before information can be retrieved, which is the case in the *DOM parser*, this approach also increases performance.

Extensible Messaging Engine As can be seen in figure 6.3, Axis2 provides a very modular architecture that allows for a variety of different implementations of *web services* as long as they adhere to certain specifications.

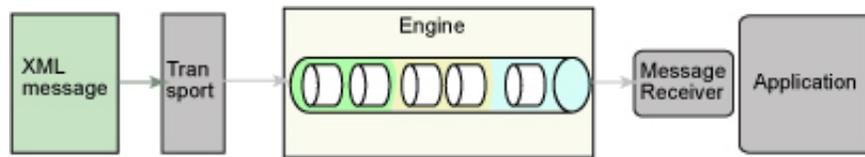


Figure 6.3. Axis2 extensibility from [16]

A variety of transports such as HTTP, SMTP, JMS and TCP can be used for message exchanges. Inside the *engine* each message goes through so-called *phases* that are part of the *piping model* which is used to implement *Message Exchange Patterns* (see section 4.6). Inside these *phases* messages can be modified, filtered or processed. The advantage of doing this inside a *phase* is that it applies to all messages. This allows for service independent processing implementations. The *message receiver* will then be re-

sponsible for handing over the actual message to the service implementation accordingly. They also take care of *synchronous* and *asynchronous* message communication.

Context Model Axis2 provides a hierarchical context model that distinguishes between the following levels:

- *Configuration* of Axis2
- *Service Group* which is a collection of *services*
- *Service* which contains several *operations*
- *Operation* that consists of *messages*
- *Message* that is sent or received

These contexts are important in the implementation of *web service* specifications such as *WS-Security* and *WS-Policy*. It means that these specifications can be applied on a level basis which provides great flexibility.

Pluggable Modules In order to provide even more flexibility and to make the implementation of *web service* specifications easier to use, Axis2 provides so-called *modules*:

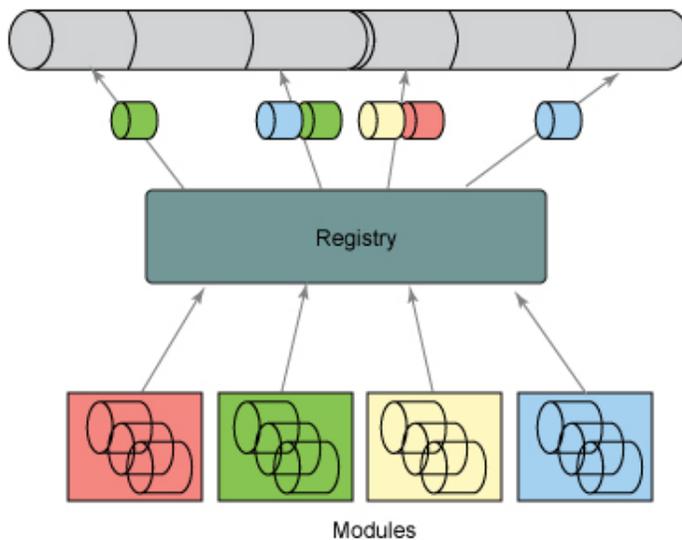


Figure 6.4. Axis2 modules from [16]

These allow an implementation of message processing that is common and useful for many *web services* to be shared. *Modules* can also be *engaged* or *disengaged* on the following levels:

- *System* which means that every service makes use of the module such as *WS-Addressing*
- *Service* which useful for *WS-Eventing*
- *Operation* that for example allows fine grained security using *WS-Security*

More information about the modules that are used in the *Transportation Security SensorNet* see section 6.1.4.

Data Binding Since a majority of data processing, element definitions and interface specifications are in XML, Axis2 provides a variety of so-called *data binding frameworks* such as XMLBeans [33], Java Architecture for XML Binding (JAXB) [29] and JiBX [80]. In addition, the *Axis2 Data Binding* (ADB) can be used, which due to its tight integration with Axis2 is highly performant. For instance, every object contains a so-called *factory* that is able to transform XML into the specific object and vice versa.

As part of this thesis further development was done by the author on this *data binding* to support a full range of *Open Geospatial Consortium* specifications such as the *Sensor Observation Service*, *Sensor Alert Service* and most notably the *Geography Markup Language*.

Several changes to the initial version of Axis2 were made in order to either fix bugs or support more functionality. In particular the build structure was adapted to work better with the *Transportation Security SensorNet* development. It makes extensive use of *Apache Ant* for the automatic generation of elements from their respective XML schema definitions, the compilation of Java classes and the deployment of *web services* and clients

6.1.1.3 SOAP

Service Oriented Architectures make use of SOAP as a flexible message format. The *Transportation Security SensorNet* does the same since *web service* specifications can easily be integrated and applied to SOAP messages.

An in-depth discussion of SOAP can be found in section 4.2.

6.1.1.4 WSDL

All services in the *Transportation Security SensorNet* are defined using the *Web Services Description Language* (WSDL) version 2.0. An in-depth introduction is provided in section 4.5. This section explains how the combination of WSDL files and XML schemas make up the foundation of a web service.

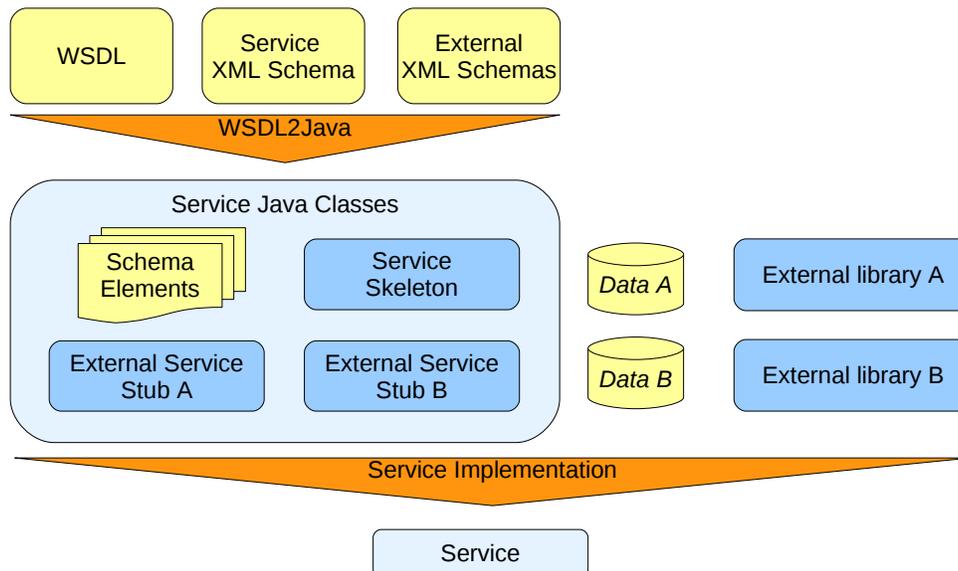


Figure 6.5. Service composition

Utilizing the automatic code generator of Axis2 called *WSDL2Java*, all elements defined in the XML schemas are available as Java classes. Furthermore a *skeleton* is created that contains the operations of the web service as methods. Interaction with

other services is achieved using their respective *stubs* which provide methods for each of its defined operations. They allow clients to perform requests directly using Java. This is because Axis2 provides the entire SOAP stack from the message format to the parsing into elements all the way up to the invocation of a method that represents a service operation.

The composition of the generated parts, data and external libraries then forms the actual service implementation.

6.1.2 Services

The services that are implemented in the *Transportation Security SensorNet* make use of a variety of components. For long term information storage, a MySQL database is used. A so-called *object-relational mapping* tool called *Hibernate* [41] enables objects to be stored and retrieved transparently without the need of complicated database interactions.

Esper [27] provides complex event and alarm processing and is used at the *Virtual Network Operation Center*. The *Alarm Processor* at the *Mobile Rail Network* currently uses a less complex approach.

The *Sensor Node* is responsible for the actual communication with the sensors. It makes use of the so-called *Hi-G-Tek* (HGT) [42] protocol and a serial connection library for Java called *RXTX*.

Each component and its particular use is explained in the later sections when each individual service is described. At a high level, one of the main aspects when dealing with web services is the definition of whether they are *stateless* or *stateful*:

6.1.2.1 Stateless

By default web services are meant to be *stateless*. This is because most message exchanges are completely independent of each other. Web services usually offer calculations, information or capabilities that only require the service to perform a specific

action and give a response. This is part of the *autonomy* approach of web services (see chapter 4).

Even in the case where a web services provides data, the service is still considered *stateless* since the retrieval of the data at any given time is not dependent on the internal state of the service but only on the underlying data. If the data changes there is no state change in the web service and it still provides the same functionality.

6.1.2.2 Stateful

The need for *stateful* web services has been identified for the *Transportation Security SensorNet* because there are certain limitations in just using *stateless* web services. Given a so-called *online* data processor that analyzes sensor data; using a *stateless* web service, it is impossible to react to trends and complex events because the service is limited to single data objects that it receives.

Let us say that a web service is monitoring whether *seals* that lock cargo containers are broken and is supposed send out warning messages whenever they are. The service has limited capacity in terms of storing historic data but should still be able to intelligently determine if a sensor reading that shows that a seal is broken is just a misreading or a real threat. This is only possible if the service keeps track of previous states. In contrast, a *stateless* service would only be able to react to the current reading and is forced to make decisions based on this single piece of data.

Another example is the *Alarm Processor* service (see section 6.3.2) at the *Mobile Rail Network* that is used in the *Transportation Security SensorNet* implementation. It classifies sensor data from containers either as *information* or *security* depending on whether one is currently allowed to open the container or not.

6.1.3 Clients

Clients are able to make use of the operations provided by the *web services*. They usually utilize the same modules as the service. This means that in theory all *web*

services could have clients. Since a lot of the services in the *Transportation Security SensorNet* interact independently from users, the number of clients that are available to users is actually smaller.

One of the aspects of clients in the *Transportation Security SensorNet* is the management of the sensors. The *Sensor Management* service (see section 6.4.1) provides this among other things like retrieving the location of a particular *Sensor Node*.

Another aspect is the management of alarm notifications. For this purpose the *Alarm Reporting* service (see figure 6.13) defines various management operations for clients.

In order to facilitate the use of those clients, a so-called *Command Center Graphical User Interface* was implemented that works just like a desktop application. This is in addition to the command line interface that every client provides using the *Apache Commons Command Line Interface* (CLI) library.

6.1.4 Modules

Axis2 provides the possibility to “plug in” so-called *modules* that add functionality or change the way a service behaves. This allows a specific capability to be shared among different services without having to implement it in each of them. In general, the web service specifications that are used in Axis2 are implemented as modules. For more information see section 6.1.1.2.

6.1.4.1 Ping

In order to check the status of a particular service Axis2 provides a module that adds an operation called *pingService* to a service. This can be used to check the status of either a specific operation or all operations that the service defines. The client part that actually uses this operation was not part of Axis2 and had to be implemented by the author.

6.1.4.2 Logging

Especially for debugging purposes and performance evaluations, it is of great benefit to be able to see the raw SOAP messages that are sent and received. The so-called *logging* module that was implemented provides this functionality. In particular the following information is captured:

- *Time* when the message was sent or received
- *Service* which is used
- *Operation* that is being executed
- *Direction* of the message, which can be either incoming or outgoing. Note that there are special directions that deal with incoming and outgoing faults.
- *From* address of the message
- *Reply to* address that may differ from the *From* address
- *To* address of the message
- *Schema element* that is being “transported” as part of the operation containing the request parameters or the response elements
- *Size* of the message in bytes
- *Message* which represents the entire SOAP message in a readable form

In terms of analyzing the *Transportation Security SensorNet* and its performance the *logging* module was engaged in all services. More information on the findings can be found in chapter 7.

6.1.4.3 Addressing

An implementation of the *WS-Addressing* specification as described in section 4.3.1 comes as part of the *addressing* module in the Axis2 core. It fully supports all components of the standard and its *ReplyTo* and *RelatesTo* fields are used among other things to allow for *asynchronous* communication (see section 6.1.6) in the TSSN.

6.1.4.4 Savan

The *Savan* module enables web services and clients in Axis2 to make use of various forms of subscription mechanisms as defined by the *WS-Eventing* specification (see section 4.3.2).

6.1.4.5 Rampart

In order to provide security according to the *WS-Security* specification (see section 4.3.3) for the TSSN the *Rampart* module was developed by Axis2. It makes extensive use of the *WS-SecurityPolicy* standard described by Lawrence et al. [50].

6.1.5 Subscriptions

Subscriptions are a fundamental part of the overall architecture of the *Transportation Security SensorNet*. They are used by the *Alarm Processor* at the *Virtual Network Operation Center* as well as in the *Mobile Rail Network*. These web services, that act as information publishers, utilize the *Savan* module to provide the operations defined in *WS-Eventing*.

6.1.6 Synchronous and asynchronous communication

By default Axis2 uses request-response in a *synchronous* manner. This means that the client has to wait and is therefore *blocking* until it receives the response from the service. In certain scenarios, for instance when the service needs a large amount of processing time, the client can experience timeouts. Furthermore, in the *Transportation Security SensorNet* where the *Mobile Rail Network* is only intermittently connected to the *Virtual Network Operation Center*, *synchronous* communication shows its limitations.

A better option is to make the communication between services *asynchronous*. This resolves timeout issues and deals with connections that are only temporary. The follow-

ing aspects need to be taken into consideration when using *asynchronous* communication:

6.1.6.1 Client

The client needs to make changes in regard to the how the request is sent out. Axis2 provides a low-level *non-blocking client API* and additional methods in the service stubs that allow callbacks to be registered. These so-called *AxisCallbacks* need to implement two methods, one that is being invoked whenever the response arrives and the other to define what happens in case of an error.

6.1.6.2 Transport Level

Depending on the transport protocol that is being used, Axis2 supports the following approaches.

- *One-way* uses one channel for the request and another one for the response such as the *Simple Mail Transfer Protocol* (SMTP)
- *Two-way* allows the same channel to be used for the request and the response, for example HTTP

For asynchronous communication to work the two-way approach was modified through the Axis2 *client API* which provides the option of using a *separate listener*. This tells the service that it is supposed to use a new channel for the response. In order to correlate request and response messages Axis2 makes use of the *WS-Addressing* specification, in particular the *RelatesTo* field.

6.1.6.3 Service

The final piece of asynchronous communication is to make the service processing asynchronous as well. This is done by specifying so-called *asynchronous message receivers* in the services configuration in addition to the *synchronous* ones.

Axis2 then uses the *ReplyTo* field of the *WS-Addressing* header in the client as a sign to send an immediate *acknowledge* of the request back to it. Furthermore it processes the request in a new thread and sends the response out when it is done, allowing the communication to be performed in asynchronous manner completely.

There exist various forms of transport protocols that are suitable for *asynchronous* communication. Axis2 by default supports HTTP, SMTP, JMS and TCP as transports but other transports can easily be defined and plugged in. The *Java Message Service* (JMS), for instance, makes use of so-called *queues* which allow clients and services to store on them and retrieve messages in a flexible manner. This is essential for satellite communication which is part of the next stage of the implementation of the *Transportation Security SensorNet*.

6.2 TSSN Common Namespace

Elements are often shared among a variety of services. Since defining the same element over and over again is neither a scalable nor maintainable approach, it makes sense to specify a common namespace for them and let the web services that want to use them, include them. In the *Transportation Security SensorNet* these shared elements are part of the so-called *TSSN Common* namespace.

In particular the following elements and types are defined:

Simple Types

A *TrainID_t* represents a unique assembly unit of engines and rail cars.

The *SensorNodeID_t* uniquely identifies a *Sensor Node*.

A *HGT_SealID_t* is a combination of four characters and eight numbers that is used to identify a *Hi-G-Tek* tag or sensor.

Location

The *LocationBean* is used to store GPS location information. It consist of:

- *longitude*
- *latitude*
- *quality* of the so-called *GPS fix*

The so-called *quality* can be one of the following predefined ones:

- *none*, no position information available
- *old*, more than 1 minute without a valid position
- *poor*, last position information less than 60 seconds old and *GPS fix* is bad
- *fair*, last position information less than 40 seconds old and *GPS fix* is okay
- *good*, last position information less than 20 seconds old and *GPS fix* is okay
- *great*, last position information less than 10 seconds old and *GPS fix* is good

Messages

A *Status* is used widely as a return message and indicates the success or failure of an operation. It has the following fields:

- *status* that is defined as a boolean and signals *success* or *failure*
- *message* which contains information on the success or failure

A *Failure* that represents the occurrence of an exception is made up of a simple *message*.

Alarms

The *AlarmSeverity* which can be either one of the following:

- *Information* that someone might be interested in
- *Maintainence* related
- *Security* breach of a seal
- *Hazard* that needs to be investigated

The *AlarmType* can be one of these:

- *Message* that contains no other inherent meaning
- *SensorLimitReached* that is propagated when an observed property value exceeds certain limitations
- *SensorLost* which means that the specified sensor cannot be reached
- *SensorFound* which informs of an established connection to a particular sensor
- *Exception* that has occurred in a service

The one element that is most commonly used for the alarm notifications is the *MRN_AlarmBean* because it contains all the valuable information of an alarm.

- *SourceNode* that identifies the *Sensor Node*
- *TrainId* that identifies the associated train
- *TimeStamp* when the alarm occurred
- *Type* of the alarm, an *AlarmType*
- *Severity* of the alarm, an *AlarmSeverity*
- *Message* that contains the alarm data or information
- *Location* of the alarm, a *LocationBean*

Other commonly used or shared elements such as the *ExceptionReport* are part of the *web service* specifications and are described separately when explaining each service individually in the following sections.

6.3 Mobile Rail Network

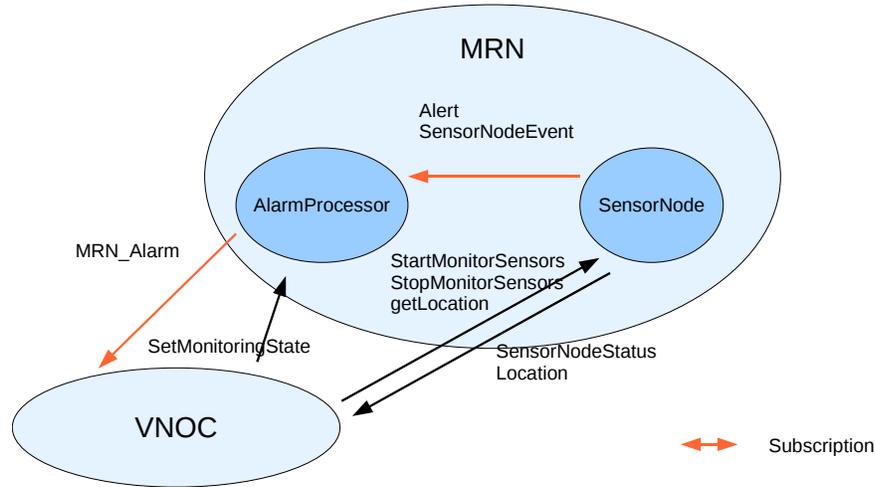


Figure 6.6. Mobile Rail Network message overview

The *Mobile Rail Network* is a collection of services that is located on a train or in a rail yard. Its services provide the abilities to manage sensors, monitor them and propagate sensor alerts to the *Virtual Network Operation Center*. This section describes them in detail.

6.3.1 Sensor Node

The *Sensor Node* contains the actual sensor monitoring and management application and its components are shown in figure 6.7. It provides several abstraction layers that allow various forms of sensors to be used. The current implementation makes use of so-called *Hi-G-Tek* (HGT) sensors. Interaction with these sensors is performed using a so-called *Automatic Vehicle Location* (AVL) reader. The *Sensor Node* implements the functionality that allows higher level management of the sensors and the data that they provide through the use of a *sensor registry*, the *sensor data* storage and *sensor data processing*.

Attaching a GPS sensor to the *Sensor Node* allows sensor events to be tagged with

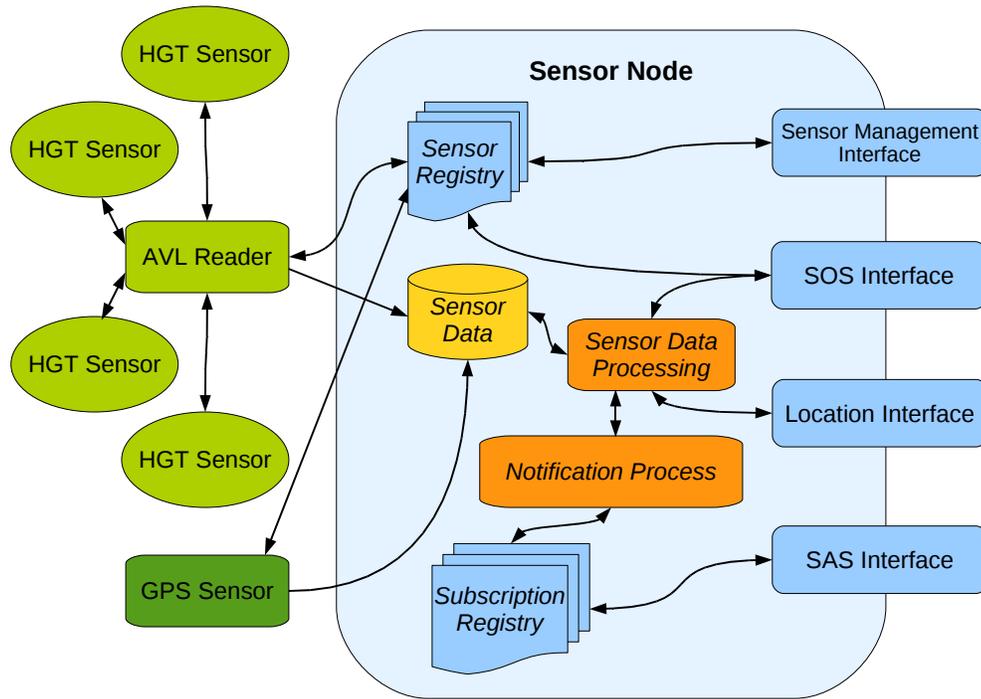


Figure 6.7. Mobile Rail Network Sensor Node

the specific location that they appeared at. The core functionality of the *Sensor Observation Service* that allows the service to offer its capabilities and observations is implemented. Furthermore, a *subscription registry* is available for alert notifications. The next sections explain the implementation details of these capabilities.

6.3.1.1 Sensor control

The following operations provide the ability to manage the underlying sensor infrastructure that is part of the *Sensor Node*.

StartMonitorSensors

The *StartMonitorSensors* operation described in table 6.1 starts the monitoring application. The *Sensor Node* then watches the status of the specified sensors identified by the *sensorIds* using the AVL reader via the HGT protocol. Note that even though

Message Exchange Pattern	In-Out
Parameters	<i>trainId</i> <i>sensorIds</i>
Response	<i>SensorNodeStatus</i>
Fault	<i>ows:ExceptionReport</i>

Table 6.1. Sensor Node StartMonitorSensors operation

the *Sensor Node* may be aware of additional sensors, it only captures events generated by the monitored sensors. The *trainId* specifies the train that the sensor node and the sensors are associated with.

StopMonitorSensors

Message Exchange Pattern	In-Out
Parameters	<i>none</i>
Response	<i>SensorNodeStatus</i>
Fault	<i>ows:ExceptionReport</i>

Table 6.2. Sensor Node StopMonitorSensors operation

The sensor monitoring application is stopped and the sensors are released by the *StopMonitorSensors* operation (table 6.2).

setSensors

Message Exchange Pattern	In-Out
Parameters	<i>none</i>
Response	<i>SensorNodeStatus</i>
Fault	<i>ows:ExceptionReport</i>

Table 6.3. Sensor Node setSensors operation

The HGT sensors that are used allow for a so-called *sleep mode*. Since they need to be “awake” in order to receive commands from the monitoring application, the *setSensors* operation described in table 6.3 sends so-called *set* signals to the sensors.

AddSeals

Message Exchange Pattern	In-Out
Parameters	<i>sensorIds</i>
Response	<i>SensorNodeStatus</i>
Fault	<i>ows:ExceptionReport</i>

Table 6.4. Sensor Node AddSeals operation

It is possible to tell the monitoring application to monitor additional sensors, which in case of HGT sensors are called seals, that are specified by the *sensorIds* using the *AddSeals* operation (table 6.4).

6.3.1.2 Location retrieval

Clients can also inquire about the current location of the *Sensor Node* when a GPS sensor has been attached.

getLocation

Message Exchange Pattern	In-Out
Parameters	<i>none</i>
Response	<i>Location</i>
Fault	<i>ows:ExceptionReport</i>

Table 6.5. Sensor Node getLocation operation

The *getLocation* operation described in table 6.5 provides a location query interface to the user. It retrieves the current location of the sensor node. Since a GPS sensor is usually attached to the *sensor node* directly, its location information is retrieved and not the one of a particular sensor.

6.3.1.3 OGC specifications

In order to provide standardized support for utilizing the functionality, the *Sensor Node* uses *WS-Eventing* to allow subscriptions to alerts that is similar to the *Sensor*

Alert Service (see section 3.2.6) and provides the following operations of the *Sensor Observation Service* (see section 3.2.5):

GetCapabilities

Message Exchange Pattern	In-Out
Parameters	<i>sos:GetCapabilities</i>
Response	<i>sos:Capabilities</i>
Fault	<i>ows:ExceptionReport</i>

Table 6.6. Sensor Node GetCapabilities operation

In accordance with the *Sensor Observation Service* specification, the *GetCapabilities* operation described in table 6.6 enables users to retrieve information about the sensors and the data they provide, the so-called *offerings*. The *Capabilities* element returned by this implementation also contains a list of sensor ids that are currently monitored.

GetObservation

Message Exchange Pattern	In-Out
Parameters	<i>sos:GetObservation</i>
Response	<i>om:Observation</i>

Table 6.7. Sensor Node GetObservation operation

The *GetObservation* operation (table 6.7) is a simplified version of the *Sensor Observation Service* equivalent and is used to retrieve current or historical sensor data from a sensor which identified by a sensor id that is part of the *GetObservation* parameter.

The provided *Observation* is a reduced version of the *Observation* in the *Observations & Measurements* specification and provides the time, format and the measurement of the sensor data observed.

The *Sensor Node* provides its functionality through the operations that were described. They allow sensor management, provide location information and OGC compliant interfaces.

6.3.2 Alarm Processor

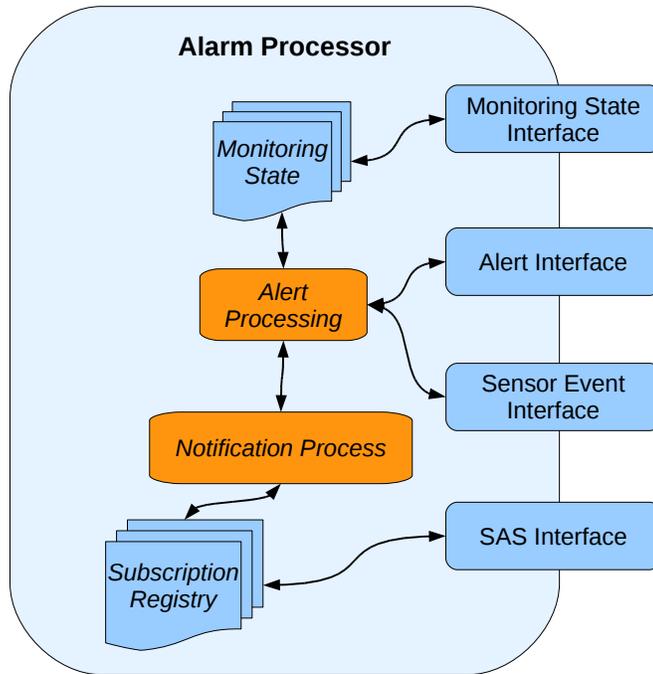


Figure 6.8. Mobile Rail Network Alarm Processor

The *Alarm Processor* on the *Mobile Rail Network* performs an initial filtering of sensor events generated by the *Sensor Node*. It subscribes to all events of the *Sensor Node*, providing interfaces for generic sensor events as well as sensor alerts. Alerts reported to the *Alarm Processor* include potential alarms that the *Sensor Node* reports, GPS acquisitions and losses, and status messages of the monitoring application such as when it is started and stopped. In case the data is not as complex as an alert, the *event* element provides a simple structure with a timestamp and a data field.

The *Alarm Processor* classifies alerts into either *information* or *security* alarms depending on its current monitoring state. It is also responsible for deciding whether or not to forward the alarm to the *Virtual Network Operation Center* for further processing and possible transmission to the decision maker. Its implementation details are discussed next.

6.3.2.1 Notifications

The following operations are defined in reference to the *Sensor Alert Service* (see section 3.2.6) for receiving notifications from the *Sensor Node*.

Alert

Message Exchange Pattern	In-Only
Parameters	<i>sas:Alert</i>

Table 6.8. Alarm Processor Alert operation

The *Alert* operation described in table 6.8 represents a simplified version of its *Sensor Alert Service* equivalent. It contains fields for storing all the necessary information about a sensor node alert. In particular:

- *SensorID* of the particular sensor causing the alert
- *TimeStamp* of the alert
- *NodeId* of the *Mobile Rail Network*
- *TrainId* that identifies the current train association
- *AlertData* which contains the raw alert information
- *Latitude* of the alert location
- *Longitude* of the alert location
- *PosQuality* that specifies the quality of the GPS signal when the location was retrieved

SensorNodeEvent

Message Exchange Pattern	In-Only
Parameters	<i>SensorNodeEvent</i>

Table 6.9. Alarm Processor SensorNodeEvent operation

Simple events can occur as well and are reported using the *SensorNodeEvent* operation (table 6.9). They contain these two fields:

- *TimeStamp* of the event
- *EventData* which contains the raw alert information

6.3.2.2 Monitoring State

The *Alarm Processor* can be configured using the following operation:

SetMonitoringState

Message Exchange Pattern	Robust-In-Only
Parameters	<i>monitoringState</i>
Fault	<i>Failure</i>

Table 6.10. Alarm Processor SetMonitoringState operation

The *SetMonitoringState* operation described in table 6.10 specifies the current monitoring state of the *Alarm Processor*. It can be used to enable or disable security. When it is enabled, seal breaks are reported using a *security* notification instead of basic *information* message.

The *Alarm Processor* uses the described operations for handling alerts and events that it receives from the *Sensor Node*. In addition, it provides functionality to specify its monitoring state, in particular to switch between *information* and *security* mode.

6.4 Virtual Network Operation Center

The *Virtual Network Operation Center* as shown in figure 6.9 represents the management facility of the TSSN and consists of services that receive and process alerts received from *Mobile Rail Networks*. It works with the *Trade Data Exchange* to associate shipment and trade information with a particular alert. Furthermore, the *Alarm Reporting* service provides clients with the ability to be notified upon specific events. The processes that are involved in performing these tasks are the topic of this section.

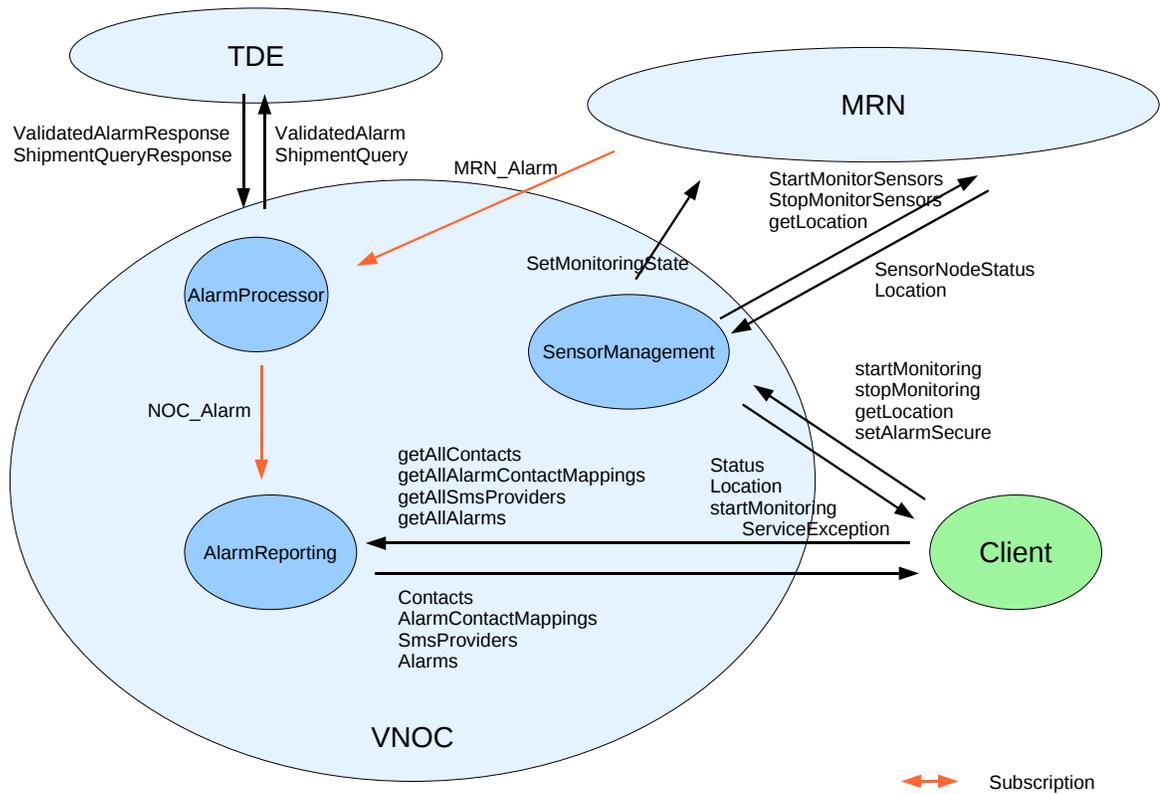


Figure 6.9. Virtual Network Operation Center message overview

6.4.1 Sensor Management

The *Sensor Management* service (figure 6.10) is responsible for controlling sensors and alarm reporting. It provides methods for starting and stopping sensor monitoring. Additionally the monitoring state which defines how alerts are interpreted and processed can be specified. The *Sensor Management* service essentially relays these “control” messages to the according *Mobile Rail Network*. Another functionality that is provided is the ability to query for a specific MRN’s location. The implementation details of the interfaces that it provides to clients are described in the following.

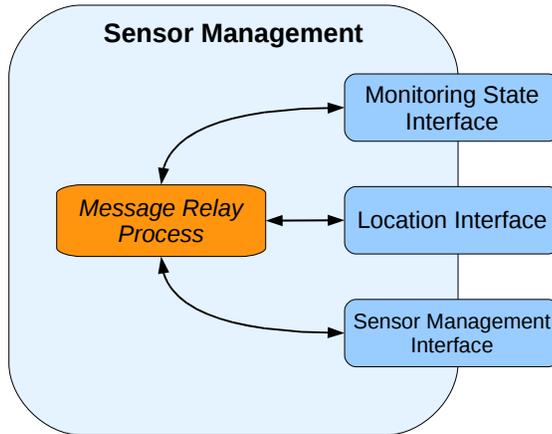


Figure 6.10. Virtual Network Operation Center Sensor Management

6.4.1.1 Sensor control

The following operations enable remote sensor management of the *Mobile Rail Networks*.

startMonitoring

Message Exchange Pattern	In-Out
Parameters	<i>collectorId</i> <i>trainId</i> <i>tagId</i> <i>sensorId</i>
Response	<i>tssn:Status</i>
Fault	<i>ows:ExceptionReport</i>

Table 6.11. Sensor Management startMonitoring operation

The *startMonitoring* operation described in table 6.11 tells the MRN that is specified by the *collectorId* to start monitoring sensors. The *collectorId* is the identifier of an individual sensor node. Furthermore the *trainId* provides the *Sensor Node* with the information of which train it is coupled to. This can be used later on to refine alarm processing and more importantly container handovers between trains. The *tagId* and *sensorId* are used in a parent-child sensor relationship. In this case a tag as the parent

would monitor the associated sensor as a child while the the sensor node only interacts with the tags. In case of sensor events the reporting chain would be sensor → tag → sensor node.

stopMonitoring

Message Exchange Pattern	In-Out
Parameters	<i>collectorId</i>
Response	<i>tssn:Status</i>
Fault	<i>ows:ExceptionReport</i>

Table 6.12. Sensor Management stopMonitoring operation

The *stopMonitoring* operation (table 6.12) is the opposite of the *startMonitoring* operation. It tells the specified sensor node to stop monitoring all sensors.

6.4.1.2 Location retrieval

Clients can inquire about the location of a particular *Sensor Node*.

getLocation

Message Exchange Pattern	In-Out
Parameters	<i>collectorId</i>
Response	<i>tssn:LocationBean</i>
Fault	<i>ows:ExceptionReport</i>

Table 6.13. Sensor Management getLocation operation

The *getLocation* operation described in (table 6.13) provides a location query interface to the user. It retrieves the current location of the specified *Sensor Node*.

6.4.1.3 Monitoring state

Alarm Processors at the *Mobile Rail Networks* can be configured using the following operations:

setAlarmSecure

Message Exchange Pattern	In-Out
Parameters	<i>collectorId</i> <i>secure</i>
Response	<i>tssn:Status</i>
Fault	<i>ows:ExceptionReport</i>

Table 6.14. Sensor Management setAlarmSecure operation

The specified MRN *Alarm Processor* can be contacted using the *setAlarmSecure* operation (table 6.14) in order to enable or disable security in its monitoring state. When the security state is enabled, seal breaks are reported using a *security* notification instead of basic *information* message.

setAlarmProcessorMonitoringState

Message Exchange Pattern	In-Out
Parameters	<i>collectorId</i> <i>monitoringState</i>
Response	<i>tssn:Status</i>
Fault	<i>ows:ExceptionReport</i>

Table 6.15. Sensor Management setAlarmProcessorMonitoringState operation

The *setAlarmProcessorMonitoringState* operation described in table 6.15 provides a more flexible configuration interface to the *Alarm Processor* on the MRN. Settings are specified in a descriptive and extensible monitoring state bean which could hold additional state information such as time frames for monitoring sensors or GPS location zones in which to automatically switch into security state. This state bean is used for instance by the *setAlarmSecure* operation.

The operations described allow the *Sensor Management* service to control *Sensor Nodes* and their monitoring state. Additionally, it is able to retrieve the location of *Sensor Nodes*.

6.4.2 Alarm Processor

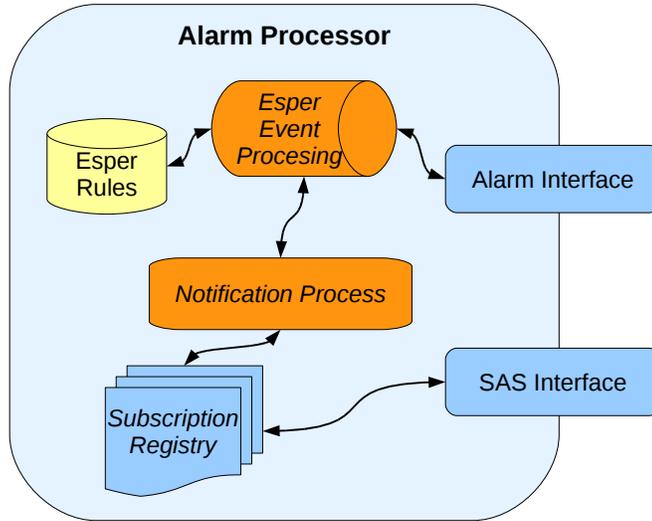


Figure 6.11. Virtual Network Operation Center Alarm Processor

In contrast to the “basic” processing that is performed by the *Alarm Processor* at the *Mobile Rail Network*, the *Alarm Processor* as shown in figure 6.11 at the VNOc has more resources such as the associated shipment and trade information available which is provided by the *Trade Data Exchange* and can therefore process alarms in a more complex way. This advanced filtering and processing is done using a complex event processing system called *Esper* developed by Bernhardt and Vasseur [7].

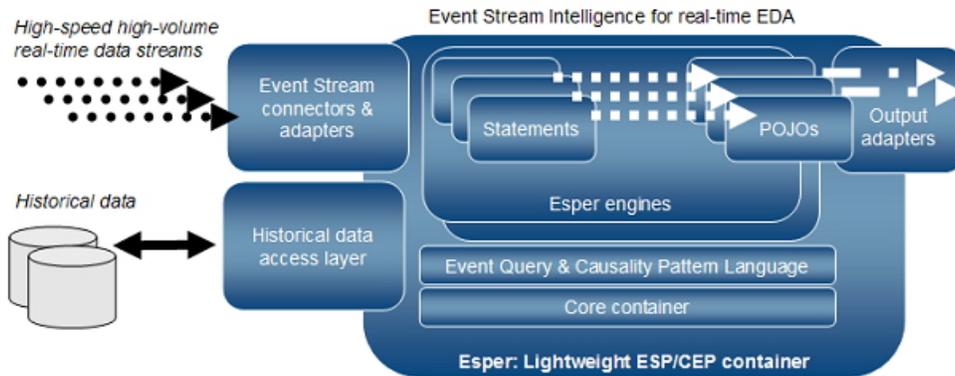


Figure 6.12. Esper architecture from [27]

Esper works on the basis of *sliding windows* in which events that are close together on the time axis are analyzed and correlated. It also supports using historical data from a variety of sources. An efficient query and filtering language called *Event Processing Language* allows for the most complex scenarios to be implemented. In the TSSN it is used for instance to filter out alarms for which shipment information could not be retrieved from the TDE and mark them as *security* notifications.

6.4.2.1 Notifications

The *Alarm Processor* receives alarm notifications from the *Mobile Rail Network* using the following operation:

MRN_Alarm

Message Exchange Pattern	In-Only
Parameters	<i>mrnpub:MRN_Alarm</i>

Table 6.16. Alarm Processor MRN_Alarm operation

The *MRN_Alarm* operation described in table 6.16 is used as a notification interface for alarms from the *Alarm Processor* on the MRN. The *Alarm Processor* service subscribes to alarms from its counterpart on the *Mobile Rail Network*. The alarms are of type *tssn:MRN_AlarmBean* (see section 6.2).

Upon receiving an alarm, shipment data is retrieved from the *Trade Data Exchange* and attached to the original alarm. *Esper* then processes the alarm and passes it on to the *Alarm Reporting* service.

The *Alarm Processor* at the VNOc primarily provides functionality for the *Mobile Rail Network* to deliver alert notifications. It uses *Esper* to perform complex event processing, taking into consideration alert data and information from the TDE, and to forward alarms to the *Alarm Reporting* service.

6.4.3 Alarm Reporting

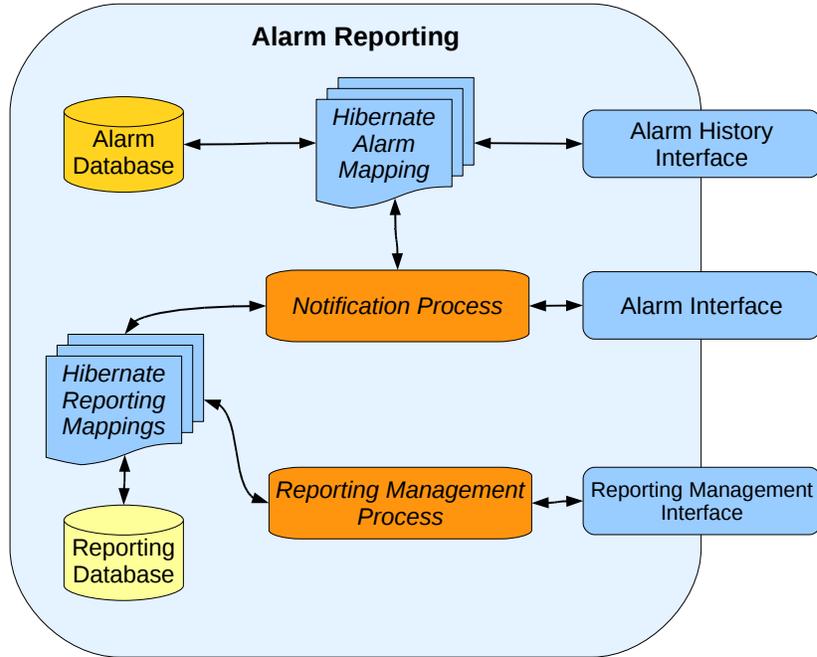


Figure 6.13. Virtual Network Operation Center Alarm Reporting

The *Alarm Reporting* service deals with the following two aspects. First, it stores alarms long term to allow for in-depth reporting and analysis. Second, clients that want to be notified of particular alarms can register with the *Alarm Reporting* service. Whenever alarms occur notifications are sent out to the registered clients via email and/or SMS accordingly.

For long term data storage and to maintain a registry of the client notifications the *Alarm Reporting* service makes use of the *MySQL* database. In order to remain flexible and provide an abstraction layer to the core database functionality a tool called *Hibernate* [41] was utilized. An excellent introduction to the so-called *object-relational* mapping is provided by Bauer and King [3]. The main advantage is that objects referenced in code can easily be *persisted* into a relational database and vice versa. The only thing that needs to be defined is the so-called *mapping*. Once that has been defined

Hibernate takes care of the rest.

Since the objects that are being stored in the database are defined using XML schemas and then automatically compiled into Java objects during the build process, it makes sense to specify the mappings in XML as well. This is done in the *Transportation Security SensorNet*. Another approach that is supported by *Hibernate* is using so-called *annotations* within the Java objects themselves. This is not possible because of the aforementioned build process as the objects would have to be reannotated at every build.

The registry that is used for notifications contains so-called *alarm contact mappings* that specify what kind of alarms a specific contact wants to be notified of. In case the contact wants to receive SMS notifications, a SMS provider has to be specified as well. The implementation details of the interfaces provided are described in the following.

6.4.3.1 SMS Providers

SMS providers have the following fields:

- *id* that uniquely identifies a provider
- *name* of the provider
- *emailSuffix* which is used for the email-based delivery of sms messages

The *emailSuffix* is used to construct an email address that is used to send out the SMS. For example “123456789@sampleProvider.com” where “123456789” is the phone number of the contact and “@sampleProvider.com” the email suffix of the phone provider.

addSmsProvider

Message Exchange Pattern	In-Out
Parameters	<i>SmsProvider</i>
Response	<i>tssn:Status</i>

Table 6.17. Alarm Reporting addSmsProvider operation

The *addSmsProvider* operation described in table 6.17 adds a new sms provider to the service. Note that the sms provider id is left blank (*null*) intentionally in this case and only the name and the email suffix have to be provided. The *Alarm Reporting* service automatically assigns an id to the new sms provider and stores it.

updateSmsProvider

Message Exchange Pattern	In-Out
Parameters	<i>SmsProvider</i>
Response	<i>tssn:Status</i>

Table 6.18. Alarm Reporting updateSmsProvider operation

Within the *updateSmsProvider* operation (table 6.18) the sms provider is identified by its id. The service looks for changes made to the sms provider and saves them.

removeSmsProvider

Message Exchange Pattern	In-Out
Parameters	<i>SmsProvider</i>
Response	<i>tssn:Status</i>

Table 6.19. Alarm Reporting removeSmsProvider operation

The *Alarm Reporting* service identifies sms providers that match the provided name and email suffix with elements in the database and removes them. The *removeSmsProvider* operation described in table 6.19 allows for pattern-based removal of sms providers. It also checks if there are still contacts associated with it.

removeSmsProviderById

Message Exchange Pattern	In-Out
Parameters	<i>Id</i>
Response	<i>tssn:Status</i>

Table 6.20. Alarm Reporting removeSmsProviderById operation

Since the id uniquely identifies an sms provider it can be removed explicitly using the *removeSmsProviderById* operation (table 6.20). The same check as in the *removeSmsProvider* operation is in place.

getAllSmsProviders

Message Exchange Pattern	In-Out
Parameters	<i>none</i>
Response	<i>SmsProviders</i>

Table 6.21. Alarm Reporting getAllSmsProviders operation

The *getAllSmsProviders* operation described in table 6.21 provides an interface to retrieve all available sms providers in a list form.

6.4.3.2 Contacts

Contacts have the following fields that contain general information about them:

- *id* that uniquely identifies a contact
- *affiliation* that represents an organization or company
- *name* which usually is first and last name of a person
- *email* address of the contact
- *smsProviderId* reference to the phone provider's *email-to-SMS* service
- *cellPhoneNumber* for SMS notifications

An *email* address or *cellPhoneNumber* must be provided, not necessarily both.

addContact

Message Exchange Pattern	In-Out
Parameters	<i>Contact</i>
Response	<i>tssn:Status</i>

Table 6.22. Alarm Reporting addContact operation

The *addContact* operation (table 6.22) is similar to the *addSmsProvider* operation in the sense that no id has to be provided for the new contact. The contact is stored in the database with an automatically assigned id.

updateContact

Message Exchange Pattern	In-Out
Parameters	<i>Contact</i>
Response	<i>tssn:Status</i>

Table 6.23. Alarm Reporting updateContact operation

Within the *updateContact* operation described in table 6.23 the service retrieves the specified *contact* by its id and saves the changes that were made to it.

removeContact

Message Exchange Pattern	In-Out
Parameters	<i>Contact</i>
Response	<i>tssn:Status</i>

Table 6.24. Alarm Reporting removeContact operation

The *removeContact* operation (table 6.24) removes the specified contact. It also allows for pattern based removal. A check is in place that prevents removal of contacts for which there still exist alarm contact mappings.

removeContactById

Message Exchange Pattern	In-Out
Parameters	<i>Id</i>
Response	<i>tssn:Status</i>

Table 6.25. Alarm Reporting removeContactById operation

The contact that is identified by the id is removed using the *removeContactById* operation described in table 6.25. The same check as in *removeContact* is in place.

getAllContacts

Message Exchange Pattern	In-Out
Parameters	<i>none</i>
Response	<i>Contacts</i>

Table 6.26. Alarm Reporting getAllContacts operation

A list of all the defined contacts can be retrieved with the *getAllContacts* operation (table 6.26).

6.4.3.3 Alarm Contact Mappings

Alarm contact mappings have the following fields:

- *id* that uniquely identifies a mapping
- *severity* of the alarm
- *type* of alarm
- *contactId* which references a particular contact
- *method* of notification (email or SMS)

These mappings are used by the *Alarm Reporting* service to determine what kind of notifications each contact receives and which *methods* to use for delivering them.

addAlarmContactMapping

Message Exchange Pattern	In-Out
Parameters	<i>AlarmContactMapping</i>
Response	<i>tssn:Status</i>

Table 6.27. Alarm Reporting addAlarmContactMapping operation

A new “alarm to contact” mapping is created using the defined entities with the *addAlarmContactMapping* operation (table 6.27).

updateAlarmContactMapping

Message Exchange Pattern	In-Out
Parameters	<i>AlarmContactMapping</i>
Response	<i>tssn:Status</i>

Table 6.28. Alarm Reporting updateAlarmContactMapping operation

Within the *updateAlarmContactMapping* operation described in table 6.28 the service retrieves the specified *alarm contact mapping* by its id and saves the changes that were made to it.

removeAlarmContactMapping

Message Exchange Pattern	In-Out
Parameters	<i>AlarmContactMapping</i>
Response	<i>tssn:Status</i>

Table 6.29. Alarm Reporting removeAlarmContactMapping operation

The *removeAlarmContactMapping* operation (table 6.29) removes the specified alarm contact mapping.

removeAlarmContactMappingById

Message Exchange Pattern	In-Out
Parameters	<i>Id</i>
Response	<i>tssn:Status</i>

Table 6.30. Alarm Reporting removeAlarmContactMappingById operation

The alarm contact mapping that is defined by the id is removed using the *removeAlarmContactMappingById* operation described in table 6.30.

getAllAlarmContactMappings

The service provides a list of all the alarm contact mappings that are in place with the *getAllAlarmContactMappings* operation (table 6.31).

Message Exchange Pattern	In-Out
Parameters	<i>none</i>
Response	<i>AlarmContactMappings</i>

Table 6.31. Alarm Reporting getAllAlarmContactMappings operation

6.4.3.4 Notifications

The *Alarm Reporting* service receives alarm notifications from the *Alarm Processor* at the *Virtual Network Operation Center* using the following operation:

NOC_Alarm

Message Exchange Pattern	In-Only
Parameters	<i>nocpub:NOC_Alarm</i>

Table 6.32. Alarm Reporting NOC_Alarm operation

This operation is used to provide a notification interface primarily for the subscription of alarms from the *Alarm Processor*. The *Alarm Reporting* service subscribes to alarms and provides this operation for its notifications. An alarm here is a combination of the *tssn:MRN_AlarmBean* and shipment and trade information received from the *Trade Data Exchange*.

6.4.3.5 Alarm history

getAllAlarms

Message Exchange Pattern	In-Out
Parameters	<i>none</i>
Response	<i>Alarms</i>

Table 6.33. Alarm Reporting getAllAlarms operation

A list of all the alarms that the service has received are retrieved using the *getAllAlarms* operation described in table 6.33. The alarms are of type *tssn:MRN_AlarmBean*. Note that the associated shipment data is not stored in the *Alarm Reporting* service as

it is permanently available in the *Trade Data Exchange*.

6.5 Trade Data Exchange

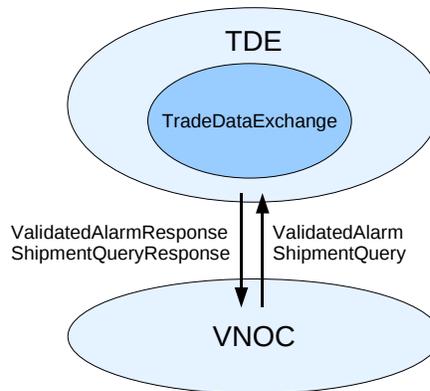


Figure 6.14. Trade Data Exchange message overview

The *Trade Data Exchange* [79], as shown in figure 6.14, in a sense represents a shipment and other trade data information provider. It aims to be a collection of heterogeneous systems that stores and manages the business aspects of a transport of goods. This is due to the fact that there is a variety of different systems implemented by the parties that participate in the transport chain (see section 2.1 and section 2.3). Some provide route information while others manage contracts and shipment data. For the current implementation of the *Transportation Security SensorNet* this “collection” of information and management services is combined into a single service, the *Trade Data Exchange* service.

6.5.1 Trade Data Exchange Service

The *Trade Data Exchange* service (figure 6.15) interacts with the *Alarm Processor* at the *Virtual Network Operation Center*. Upon request it provides shipment and trade information for a specified alarm. It also provides functionality that can be used for long term alarm storage, although in its current implementation fairly limited. Since

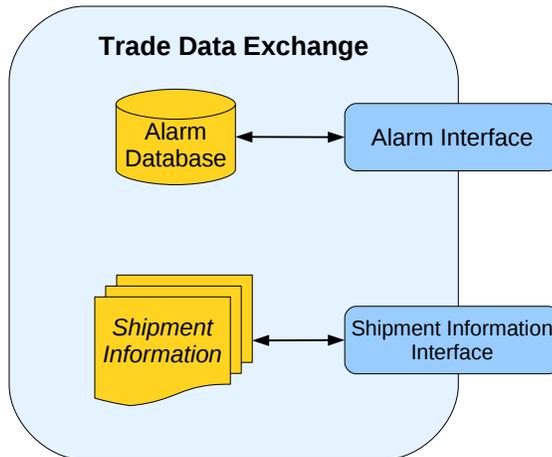


Figure 6.15. Trade Data Exchange Service

the service was designed externally, the elements used are not compatible to the TSSN common elements or any of the other services.

The *alarm data* element used has the following fields:

- *timeOccured* which represents the time when the alarm occurred
- *train_id* that uniquely identifies a train
- *tag_id* that uniquely identifies a tag (in this case a seal)
- *sensor_id* that uniquely identifies a sensor
- *alarm_type* which is either *Door open*, *Door closed*, *Sensor missing* or *Sensor returned*

This element has some shortcomings such as no location information, no alarm data field and limited *alarm types* but is currently used by the TSSN for the lack of a better interface to the shipment information.

6.5.1.1 Information inquiry

The following operation is provided to retrieve shipment and trade information from the *Trade Data Exchange*.

ShipmentQuery

Message Exchange Pattern	In-Out
Parameters	<i>alarm_data</i>
Response	<i>shipment_data</i>

Table 6.34. TradeDataExchange ShipmentQuery operation

The *ShipmentQuery* operation as described in table 6.34 provides *shipment_data* for the specified *alarm_data*. The shipment data contains the following information:

- *train_id* that uniquely identifies a train
- *equipment_id* that uniquely identifies a rail car; it consists of an *initial* and a *number*
- *car_position* of the container that the sensor is attached to
- *bic_code* that uniquely identifies a so-called *intermodal unit*
- *stcc* which is the *Standard Transportation Commodity Code* of the goods shipped

It has to be noted that no route information is made available through this inquiry.

6.5.1.2 Alarm storage

For long term storage of alarms the next operation is provided:

ValidatedAlarm

Message Exchange Pattern	In-Out
Parameters	<i>alarm_data</i>
Response	<i>status</i>

Table 6.35. TradeDataExchange ValidatedAlarm operation

Using the *ValidatedAlarm* operation (table 6.35) the *Trade Data Exchange* service receives *alarm_data* and stores it in a database.

6.6 Open Geospatial Consortium Specifications

As described before, the amount of work that is required to fully implement specifications of the *Open Geospatial Consortium* such as the *Sensor Observation Service* and the *Sensor Alert Service* is immense. The focus of the first stage of the implementation of the *Transportation Security SensorNet* is on the sensor management and alarm notification capabilities. However, at the *Mobile Rail Network* the *Sensor Node* provides an implementation for the *Sensor Observation Service* as defined by the OGC. Furthermore, services in the TSSN that utilize subscriptions, in particular the *Alarm Processor*, are able to receive *subscribe* requests and publish *alerts* in a manner that is similar to the *Sensor Alert Service*. The difference to the proposed SAS specification is that the services that subscribe are already aware of the *capabilities*, *sensor* types and *alert* types. Therefore the operations that allow the retrieval of this information, as described in section 3.2.6, need to be implemented in order to be fully compliant.

Chapter 7

Implementation Results

In this chapter tools that were developed and used to monitor the *Transportation Security SensorNet* are described. The *logging* module (section 7.1) plays the most important part as it captures message flows throughout the TSSN. These can then be analyzed using the *log parser* (section 7.2) and visualized by the *Visual SensorNet* tool (section 7.3). Performance measurements that were made throughout a series of trials are the used to evaluate the communication speed, processing times and alarm notifications (see section 7.4) within the TSSN.

7.1 Logging Module

The *logging* module as described in section 6.1.4.2 provides extensive logging capabilities to the *web services* in the *Transportation Security SensorNet*. It was *engaged* during development and testing of the entire system since it logs all messages that are sent and received. In addition, it also writes the raw contents of the SOAP messages into log files.

7.2 Log Parser

The *log parser* enables parsing and most importantly the merging of log files. It transforms the raw SOAP messages back into Java elements that can then be filtered and analyzed.

7.2.1 Abstraction Layer Model

Since SOAP is essentially XML, information from the so-called *log messages* can be retrieved using *XPath* [8] *path expressions*. For this purpose the *log parser* provides an object *abstraction layer model* that corresponds to the specific parts in the SOAP message.

An example mapping is shown in figure 7.1. It displays the structure of the original SOAP message (for more information on SOAP see section 4.2) on the left and the equivalent *log parser* objects on the right. Note that the corresponding objects highlighted in yellow are actual classes while the *Header* and *Body* are not abstracted separately.

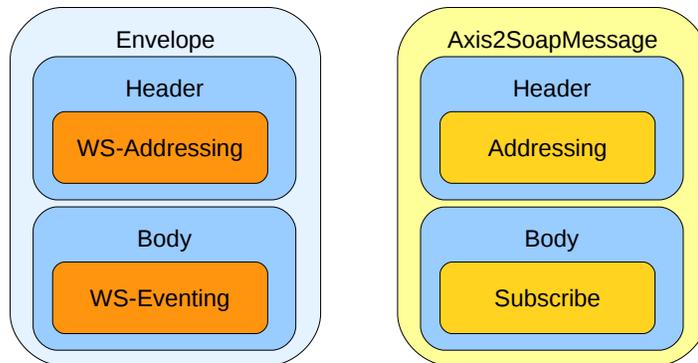


Figure 7.1. SOAP message (left) to Log parser classes (right) comparison

The *log parser* objects would then provide access to their properties using *XPath expressions*. In this case they correspond to their respective *web service* specifications but they could also be defined according to the XML schema definitions of any other

element. For example, for the *WS-Addressing* (see section 4.3.1) equivalent object the *path expressions* in table 7.1 are used:

XPath expression	Method equivalent
//To/text()	getTo()
//ReplyTo/Address/text()	getReplyTo()
//From/Address/text()	getFrom()
//MessageID/text()	getMessageId()
//RelatesTo/text()	getRelatesTo()
//Action/text()	getAction()

Table 7.1. XPath expressions for *WS-Addressing*

This mapping process is easily defined and allows for an in-depth analysis of the messages that are sent and received in the *Transportation Security SensorNet*.

7.2.2 Message Types

Since the *logging* module is enabled on both ends of a message exchange, the *log parser* is able to correlate messages. In order to do this it makes use of the so-called *message id* that is provided by the *WS-Addressing* specification. The following two types of message associations are present in the log files:

Transmit-Receive Pair Whenever a message is sent out by a particular client or service it is captured by the *logging* module. The receiving service logs the message as well but as an incoming message. The content of the message is essentially the same which can also be seen by the fact that they have the same *message id*. The outgoing and the incoming message are combined and form what is called a *transmit-receive pair*.

This allows us to compute the message transfer or so-called *transmit* time which describes how long it takes to transmit the message from one entity to another using the following equations:

$$transmitTime_1 = time_{2.Incoming} - time_{1.Outgoing} \quad (7.1)$$

$$transmitTime_2 = time_{4.Incoming} - time_{3.Outgoing} \quad (7.2)$$

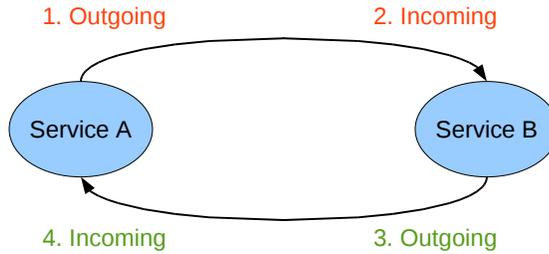


Figure 7.2. Two transmit-receive pairs (red and green)

As shown in figure 7.2 the *log parser* automatically detects the *transmit-receive pairs* and stores them in a particular list for further analysis.

Message Couple The most common *message exchange pattern* as described in section 4.6 is the *In-Out* pattern. It defines request-response based message transfers which the *log parser* calls *message couples*. A single *message couple* consists of two messages, the outgoing request and the outgoing response on the receiving entity, which is shown in figure 7.3. They can be correlated using the *WS-Addressing* specification. The request will carry a *message id* and the response a so-called *relatesTo id* in addition to its own unique *message id*.

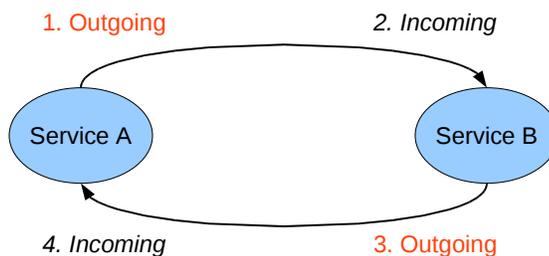


Figure 7.3. A message couple (red)

Note that a *message couple* can also be seen as a combination of two *transmit-receive pairs*. This relationship is extremely useful in computing measures such as *round trip*

and *processing* times:

$$\text{roundTripTime} = \text{time}_{4.Incoming} - \text{time}_{1.Outgoing} \quad (7.3)$$

$$\text{processingTime} = \text{time}_{3.Outgoing} - \text{time}_{2.Incoming} \quad (7.4)$$

The *log parser* provides functionality to associate messages and analyze complete end-to-end message flows. More details on the performance measurements and test results can be found in section 7.4.

7.3 Visualization

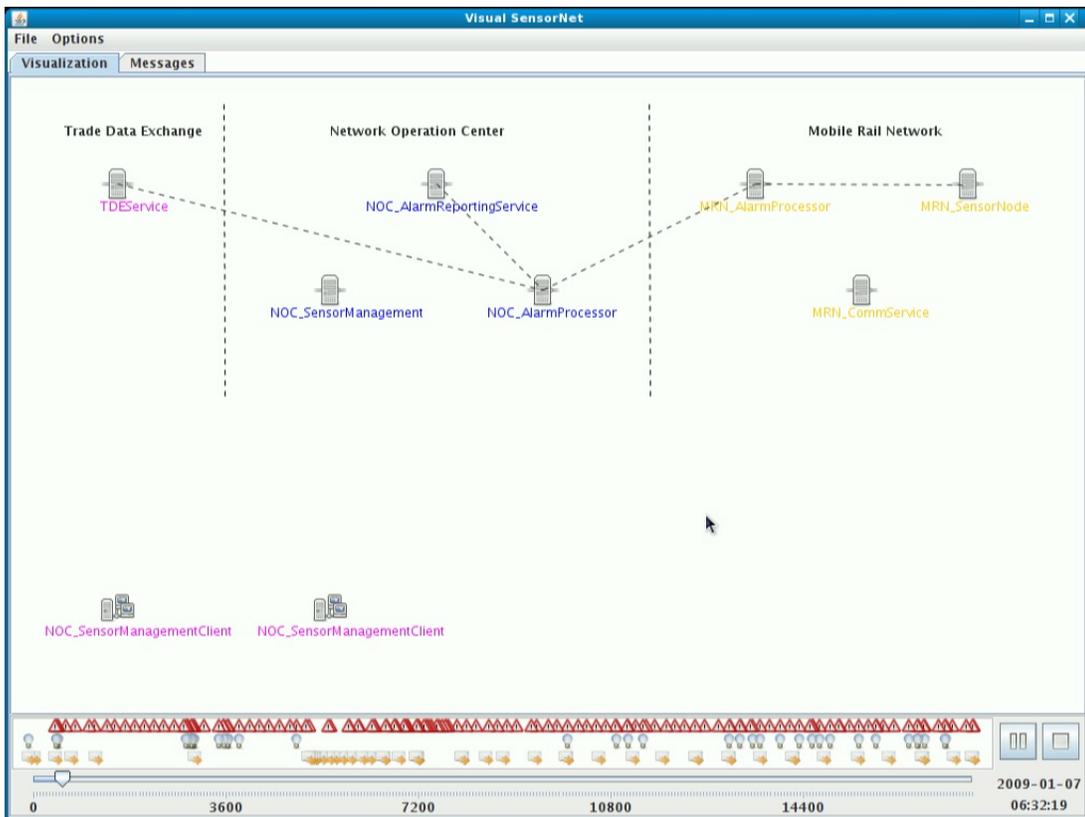


Figure 7.4. Log file and service interaction visualization

In order to be able to understand the message flows better without needing too

much of a technical background, a visualization tool called the *Visual SensorNet* was developed. It makes use of the *log parser* to display services, clients and messages that are present in log files.

The user is able to load and merge log files to create a visualization of services and clients as shown in figure 7.4. The layout of these services is defined according to their membership in a particular *service cloud*. Furthermore, any point in time that is part of the log files can be “jumped to” using the *time line*. It displays significant events in the log files:

- *Alarms, alerts* and *sensor node events* with a warning sign
- *Requests* such as location retrieval with a light bulb sign
- *Control messages* such as *start monitoring* with a message sign

The scenario that was captured by the log files can also be played back in portions or in its entirety. Using the *Visual SensorNet* tool, it is therefore possible to analyze service interactions and message flows conveniently.

7.4 Performance and Statistics

An in-depth analysis of the real world scenarios that were performed to test the *Transportation Security SensorNet* is given by Fokum et al. [31]. For the tests the *Trade Data Exchange* was deployed in Overland Park, the *Virtual Network Operation Center* at the *University of Kansas* in Lawrence and the *Mobile Rail Network* either on a truck or on a train. Note that in both cases the communication between the *Mobile Rail Network* and the *Virtual Operation Center* was established using a *GSM modem*. The main findings are as follows:

7.4.1 Road Tests with Trucks

During the tests the overall system had to deal with several issues. The location was not always available due to loss of so-called *GPS fixes*. This caused some alarms to

be reported with an inaccurate or old location. Furthermore, at some point the GSM connection broke down but could be reestablished. Note that no messages were lost in the process though.

In order to test the range of the *AVL Reader*, one of the goals was to find out at what point the reader loses contact to the sensors that it monitors. During the testing this distance was found to be about 400 meters. This was mainly due to significant hardware tuning and enhancements that were made by members of the *SensorNet* project. One of the reasons why range is so important is the fact that in the second stage the *Transportation Security SensorNet* was deployed in the engine of a train and it had to monitor sensors that were positioned on different railcars. In contrast to many other sensor networks where sensors surround a so-called *base station* in a circular manner with the aims of minimizing distance, the rail scenario represents an almost linear sensing approach where the distance to the *base station* increases for each sensor.

Another problem was the significant clock drift on the *Mobile Rail Network* during relatively short tests (about 2 1/2 hours). Unfortunately this makes some time measurements unreliable, in particular those in between the MRN and the VNOC. Note that this is not such a big problem within the *Mobile Rail Network* and *Virtual Network Operation Center service clouds* though, since there is a greater interest in relative times such as the *processing* time of an operation. This problem could partially be solved by letting the *log parser* that was used for the analysis apply a time adjustment parameter. A better and more natural solution to this problem is discussed in section 8.2.

Note that these observations are mostly hardware related. The implementation of the *Transportation Security SensorNet* as described in this thesis worked and was able to provide the sensor management as well as complete end-to-end alarm notification capabilities.

7.4.2 Short Haul Rail Trial

This more advanced scenario was performed by deploying the *Mobile Rail Network* on a locomotive of a train along with sensors attached to containers for it to monitor. The train traveled approximately 35 kilometers during the trip, from a rail *intermodal facility* to a rail yard.

The system faced some of the same issues as during the truck trials such as loss of GPS, GSM and sensor communication. The data that was collected however shows that again the *Transportation Security SensorNet* was able to deal with them and send out alarm notifications reliably. The log files were analyzed using the *log parser* and led to the following:

Message Counts An overview of the message flow is shown in figure 6.1. During the *short haul rail trial* the *Sensor Node* reported 546 alerts to the *Alarm Processor*. After filtering, the details of which are explained in section 6.3.2, 131 alarms were sent to the *Alarm Processor* at the *Virtual Network Operation Center*. For 63 of them, shipment information was queried from the *Trade Data Exchange* and 33 were stored as so-called *validated alarms*. All of the 131 alarms that the *Alarm Processor* received were sent out to *Alarm Reporting* service which notified the according contacts via SMS and email.

There were also 30 inquiries for the location of the *Mobile Rail Network*.

Message Sizes Looking at the communication between the *Virtual Network Operation Center* and the *Mobile Rail Network* one can notice the following pattern. So-called *control messages* such as *startMonitoring* or *getLocation* are always initiated at the *Virtual Network Operation Center*. Since these messages usually transmit only a small functional request, the average message size is around 690 bytes. On the other hand, Alarms are always sent from the *Mobile Rail Network* and contain of a lot of valuable information. Hence the average message size is about 1420 bytes.

Request Performance As shown in figure 7.5, the time it took for messages from the *Virtual Network Operation Center (Sensor Management)* to send requests to the *Mobile Rail Network* (either *Sensor Node* or *Alarm Processor*) and receive a response was about 4.4 seconds on average. The fastest request was answered in 0.9 seconds while the slowest took about 11 seconds.

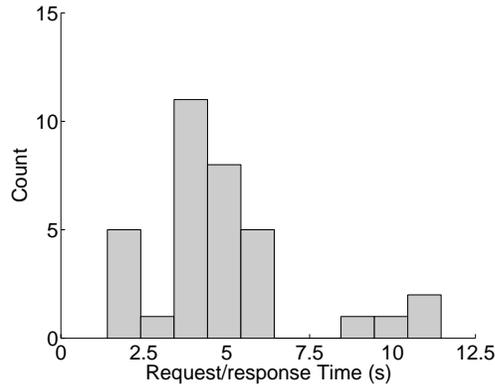


Figure 7.5. Request performance from [31]

Overall these numbers meet the expectations of the transportation industry. Performing a location inquiry given an average train speed of 30 km/h and 60 seconds to retrieve the location, the actual position and the reported one may differ by as much as 500 meters. However, the *Transportation Security SensorNet* provides location information in less than 5 seconds resulting in a maximum difference of just 41.7 meters.

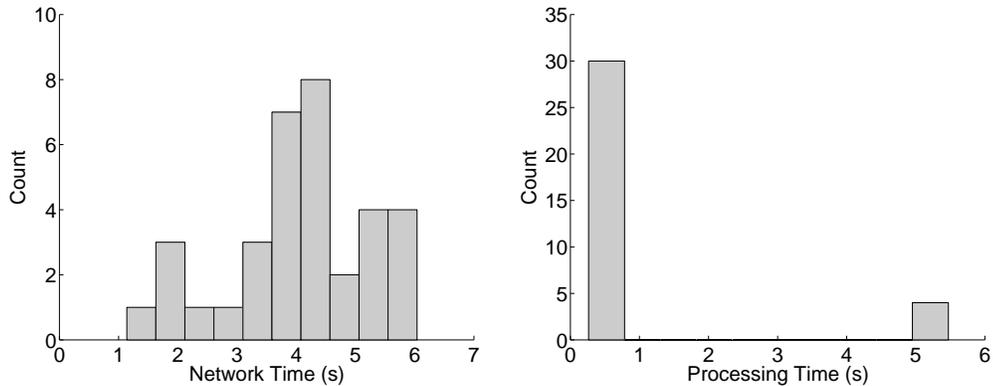


Figure 7.6. Network transmission and processing performance from [31]

The bottleneck here is the message *transmit* time as defined in equation 7.1. As shown in figure 7.6, processing on the *Sensor Node* took only 0.6 seconds on average whereas about 85% of the time is spent on message transmission. This percentage is likely to increase when switching to satellite communication instead of communicating with the GSM modem which was used in the trials.

Alarm Notification Performance Because of the problems with the clock drift, the measured times for messages coming from the *Mobile Rail Network* going to the *Virtual Network Operation Center* are unreliable. However, taking our previous findings about the request performance the time for this particular transmission can be estimated using the average *round trip* and the *processing* times:

$$\overline{transmitTime} = \frac{\frac{1}{n} \sum_{i=1}^n (roundTripTime_i - processingTime_i)}{2} \quad (7.5)$$

$$= \frac{4.4 \text{ seconds} - 0.6 \text{ seconds}}{2} \quad (7.6)$$

$$= 1.9 \text{ seconds} \quad (7.7)$$

Given this estimate, we can compute the total time it takes from for an alarm to go through the entire TSSN as shown in figure 7.7.

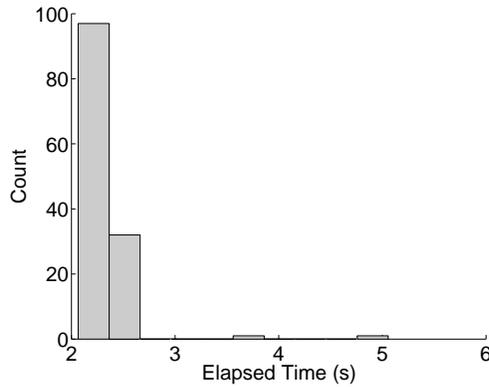


Figure 7.7. System alarm notification performance from [31]

This includes the times from the *Sensor Node* to the *Alarm Processor* at the *Mobile Rail Network*, the approximated transmit time of 1.9 seconds, and the time from the *Alarm Processor* to the *Alarm Reporting* service at the *Virtual Network Operation Center*. On average this yields about 2.1 seconds with the fastest time being just over 1.9 seconds and the slowest around 4.9 seconds.

Both, the road test with trucks and the short haul rail trial can be called successful because they displayed the capabilities of the TSSN, its good performance and that the functionality implemented in the web services worked. In particular, two of its main capabilities, location inquiry and alarm notification were extensively demonstrated. Furthermore, the time it took from registering alerts, propagating them through the *Transportation Security SensorNet* and sending out notifications accordingly is under 5 seconds and significantly smaller than expected for such a complex system.

Chapter 8

Conclusion

8.1 Current Implementation

The implementation of the *Transportation Security SensorNet* using a *Service Oriented Architecture* works. Testing has been completed in a lab environment as well as in the real world and TSSN was evaluated in chapter 7.

The complete system provides a *web services* based sensor management and alarm notification infrastructure that is built using open standards and specifications. Particular functionality within the system has been implemented in *web services* that provide interfaces according to their respective *web service* specifications.

Using standards from the *Open Geospatial Consortium* allows the integration of the system into *Geographic Information Systems*. Although not all the interfaces are fully implemented as of summer 2009, the basic *Sensor Observation Service* and *Sensor Alert Service* are. Other *Open Geospatial Consortium* specifications can be integrated a lot easier now because enhancements to the Axis2 schema compiler have been made by the author (see 6.1.1.2).

WS-Eventing plays an important role in the *Transportation Security SensorNet* as it is essential for the alarm notification chain. The specification that is used by all the clients and services is *WS-Addressing*. Note that HTTP, which represents the underlying

transport layer of most the *web services*, already provides an addressing scheme. This however, is not as useful as it seems because web services may change their *transport layer* and messages sometimes require complex routing. The reasoning behind this and other things have been explained in detail in section 4.3.1.

Overall the *Transportation Security SensorNet* provides a *Service Oriented Architecture for Monitoring Cargo in Motion Along Trusted Corridors* based on the extensible infrastructure of the *Ambient Computing Environment* for SOA. This *web services* based implementation allows for platform and programming language independence and offers compatibility and interoperability.

The integration of *Service Oriented Architecture*, *Open Geospatial Consortium* specifications and sensor networks is complex and difficult. As described in section 5.7, most systems and research focuses either on the combination of SOA and OGC specifications or on OGC standards and sensor networks. However, the *Transportation Security SensorNet* shows that all three areas can be combined and that this combination provides capabilities to the transportation and other industries that have not existed before. In particular, web services in a mobile sensor network environment have always been seen as slow and producing a lot of overhead. The TSSN, as shown by the results in chapter 7, demonstrates that this is not necessarily true.

Furthermore, the *Transportation Security SensorNet* and its *Service Oriented Architecture* allow sensor networks to be utilized in a standardized and open way through web services. Sensor networks and their particular communication models led to the implementation of asynchronous message transports in SOA and are supported by the TSSN.

8.2 Future work

After evaluating the current implementation, several points of improvement were identified.

Clock Synchronization In order to deal with the clock drift issue mentioned in section 7.4, enhancements are currently developed that will allow the time on the *Mobile Rail Network* to be adjusted using a local *Network Time Protocol* server. It is provided the so-called *pulse per second* from a GPS sensor attached to the *Sensor Node*. As a result of this there should hardly be any time synchronization problems left.

Service Discovery Due to several problems in the specific implementation of the UDDI that was used, for the trials most of the services were made aware of the other services through the means of configuration instead of service discovery. Since using a UDDI provides far better scalability, it is an essential piece of future versions of the *Transportation Security SensorNet*

Multiple service clouds During the trials all services were unique which in an operational system this is not the case. There are issues that need to be explored in dealing with multiple versions not only of single *web services* but multiple *Virtual Network Operation Centers* and *Mobile Rail Networks*. This is especially important when it comes to managing policies and subscriptions properly.

Security The current system only provides entry points for the *WS-Security* in terms of the *Rampart* module. There are several issues in the current implementation of the module, especially with regard to attaching policies to *web services* and clients. Further development is underway to implement *WS-Security*.

In between the *Virtual Network Operation Center* and the *Mobile Rail Network* communication is secured by establishing a *Virtual Private Network* (VPN). However, this is not practical using a satellite link because of performance reasons.

Sensors management is done at the *Sensor Node* but as of now there is no support for the secure handover to other *Sensor Nodes*. The remote management systems need to be improved in this area.

Asynchronous Communication The implementation of the *Transportation Security SensorNet* that was used during the trials made use of a “relatively” stable GSM modem connection that provided good performance and coverage. Furthermore, messages were sent in a *synchronous* manner.

In the next stage of development, the communication between the *Virtual Network Operation Center* and the *Mobile Rail Network* is done over a satellite link that is provided by a *communication service*. This means that several topics have to be addressed.

First, the current message sizes should be reduced in order to accommodate for the loss of speed. Possible optimizations have been discussed in section 4.2.3 but compression or conversion into binary formats are options as well.

Second, an enhancement that is currently being pursued and that deals better with message queuing on both ends of the communication is the switch to the *Java Message Service* as the transport. This is discussed by Easton et al. [25]. The *Java Message Service* uses so-called *Enterprise Service Bus* queues in order to send and receive messages. This allows the current implementation to work almost unmodified as the only thing that changes is the choice of transport for a few *web services* to fully support asynchronous communication.

8.3 Acknowledgment

The work for this thesis is supported by the *Office of Naval Research* through Award Number N00014-07-1-1042, *Oak Ridge National Laboratory* (ORNL) via Award Number 4000043403, and the *KU Transportation Research Institute* (KUTRI).

References

- [1] Eyhab Al-Masri and Qusay H. Mahmoud. Investigating web services on the world wide web. In *WWW '08: Proceeding of the 17th international conference on World Wide Web*, pages 795–804, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-085-2. URL <http://doi.acm.org/10.1145/1367497.1367605>.
- [2] Jean-Pierre Bardet and Amir Zand. Spatial modeling of geotechnical information using gml. *Transactions in GIS*, 13(1):p125 – 165, 20090101. ISSN 13611682. URL <http://search.ebscohost.com.www2.lib.ku.edu:2048/login.aspx?direct=true&db=aph&AN=36983054&site=ehost-live>.
- [3] Christian Bauer and Gavin King. *Hibernate in Action*. Manning, 2005.
- [4] Tom Bellwood. Rocket ahead with UDDI V3. IBM article, IBM, November 2002. <http://www.ibm.com/developerworks/webservices/library/ws-uddiv3/>.
- [5] Tom Bellwood, Luc Clement, David Ehnebuske, Andrew Hately, Maryann Hondo, Yin Leng Husband, Karsten Januszewski, Sam Lee, Barbara McKee, Joel Munter, and Claus von Riegen. UDDI Version 3.0. OASIS specification, OASIS, July 2002. <http://uddi.org/pubs/uddi-v3.00-published-20020719.htm>.
- [6] T. Berners-Lee, R. Fielding, and L. Masinter. Uniform Resource Identifier (URI): Generic Syntax. RFC 3986 (Standard), January 2005. URL <http://www.ietf.org/rfc/rfc3986.txt>.
- [7] Thomas Bernhardt and Alexandre Vasseur. Event-driven application servers, 2007. URL http://dist.codehaus.org/esper/JavaOne_TS-1911_May_11_2007.pdf.
- [8] Scott Boag, Anders Berglund, Don Chamberlin, Jérôme Siméon, Michael Kay, Jonathan Robie, and Mary F. Fernández. XML path language (XPath) 2.0. W3C recommendation, W3C, January 2007. <http://www.w3.org/TR/2007/REC-xpath20-20070123/>.
- [9] David Booth and Canyang Kevin Liu. Web services description language (WSDL) version 2.0 part 0: Primer. W3C recommendation, W3C, June 2007. "<http://www.w3.org/TR/2007/REC-wsdl20-primer-20070626>".
- [10] Mike Botts, George Percivall, Carl Reed, and John Davidson. OGC Sensor Web Enablement: Overview And High Level Architecture. OGC white paper,

- OGC, December 2007. http://portal.opengeospatial.org/files/?artifact_id=25562.
- [11] Don Box, David Ehnebuske, Gopal Kakivaya, Andrew Layman, Noah Mendelsohn, Henrik Nielsen, Satish Thatte, and Dave Winer. Simple Object Access Protocol (SOAP) 1.1. W3C note, W3C, July 2003. <http://www.w3.org/TR/2000/NOTE-SOAP-20000508/>.
 - [12] Don Box, Luis Felipe Cabrera, Craig Critchley, Francisco Curbera, Donald Ferguson, Steve Graham, David Hull, Gopal Kakivaya, Amelia Lewis, Brad Lovering, Peter Niblett, David Orchard, Shivajee Samdarshi, Jeffrey Schlimmer, Igor Sedukhin, John Shewchuk, Sanjiva Weerawarana, and David Wortendyke. Web services eventing (ws-eventing). W3C member submission, W3C, March 2006. <http://www.w3.org/Submission/2006/SUBM-WS-Eventing-20060315/>.
 - [13] Tim Bray, Richard Tobin, Dave Hollander, and Andrew Layman. Namespaces in XML 1.0 (second edition). W3C recommendation, W3C, August 2006. <http://www.w3.org/TR/2006/REC-xml-names-20060816>.
 - [14] Luis Felipe Cabrera, Christopher Kurt, and Don Box. An Introduction to the Web Services Architecture and Its Specifications. Microsoft technical article, Microsoft, October 2004. <http://msdn.microsoft.com/en-us/library/ms996441.aspx>.
 - [15] Marc Chanliau. Web Services Security: What's Required To Secure A Service-Oriented Architecture. Oracle white paper, Oracle, October 2006.
 - [16] Eran Chinthaka. Web services and Axis2 architecture. IBM article, IBM, November 2006. <https://www.ibm.com/developerworks/webservices/library/ws-apacheaxis2/>.
 - [17] Erik Christensen, Francisco Curbera, Greg Meredith, and Sanjiva Weerawarana. Web services description language (WSDL) 1.1. W3C note, W3C, March 2001. <http://www.w3.org/TR/2001/NOTE-wsdl-20010315>.
 - [18] Xingchen Chu. Open sensor web architecture: Core services. Master's thesis, University of Melbourne, Australia, 2005. <http://www.gridbus.org/reports/OSWA-core%20services.pdf>.
 - [19] Xingchen Chu, Tom Kobialka, and Rajkumar Buyya. Open sensor web architecture: Core services. In *In Proceedings of the 4th International Conference on Intelligent Sensing and Information Processing*, pages 1–4244. Press, 2006. <http://www.gridbus.org/papers/ICISIP2006-SensorWeb.pdf>.
 - [20] Simon Cox. Observations and Measurements - Part 1 - Observation schema. OGC implementation specification, OGC, December 2007. http://portal.opengeospatial.org/files/?artifact_id=22466.

- [21] Simon Cox. Observations and Measurements - Part 2 - Sampling Features. OGC implementation specification, OGC, December 2007. http://portal.opengeospatial.org/files/?artifact_id=22467.
- [22] Simon Cox. Observations and Measurements - Part 2 - Sampling Features. OGC schema, OGC, . <http://schemas.opengis.net/sampling/>.
- [23] Simon Cox. Observations and Measurements - Part 1 - Observation schema. OGC schema, OGC, . <http://schemas.opengis.net/om/>.
- [24] Michael J de Smith, Michael F Goodchild, and Paul A Longley. *Geospatial Analysis - A Comprehensive Guide to Principles, Techniques and Software Tools*. Matador, 2008. <http://www.spatialanalysisonline.com>.
- [25] Peter Easton, Bhakti Mehta, and Roland Merrick. SOAP over java message service 1.0. W3C working draft, W3C, July 2008. <http://www.w3.org/TR/2008/WD-soapjms-20080723>.
- [26] Thomas Erl. *Service-Oriented Architecture - Concepts, Technology, and Design*. Prentice Hall, 2005.
- [27] EsperTech. Esper - Event Stream and Complex Event Processing for Java. URL <http://www.espertech.com/>.
- [28] David C. Fallside and Priscilla Walmsley. XML schema part 0: Primer second edition. W3C recommendation, W3C, October 2004. <http://www.w3.org/TR/2004/REC-xmlschema-0-20041028/>.
- [29] Joe Fialli and Sekhar Vajjhala. Java architecture for xml binding (jaxb) 2.0. Java Specification Request (JSR) 222, October 2005.
- [30] Roy Thomas Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, University of California, Irvine, 2000.
- [31] Daniel T. Fokum, Victor S. Frost, Daniel DePardo, Martin Kuehnhausen, Angela N. Oguna, Leon S. Searl, Edward Komp, Matthew Zeets, Joseph B. Evans, and Gary J. Minden. Experiences from a Transportation Security Sensor Network Field Trial. ITTC Tech. Rep. ITTC-FY2009-TR-41420-11, University of Kansas, Lawrence, KS, June 2009.
- [32] Ian Foster, Carl Kesselman, Jeffrey M. Nick, and Steven Tuecke. The physiology of the grid: An open grid services architecture for distributed systems integration. In *Open Grid Service Infrastructure WG, Global Grid Forum*, June 2002. <http://www.globus.org/alliance/publications/papers/ogsa.pdf>.
- [33] Apache Software Foundation. XMLBeans, July 2008. URL <http://xmlbeans.apache.org/>.

- [34] Anders Friis-Christensen, Nicole Ostländer, Michael Lutz, and Lars Bernard. Designing service architectures for distributed geoprocessing: Challenges and future directions. *Transactions in GIS*, 11(6):p799 – 818, 20071201. ISSN 13611682. URL <http://search.ebscohost.com.www2.lib.ku.edu:2048/login.aspx?direct=true&db=aph&AN=28048261&site=ehost-live>.
- [35] Jesse James Garrett. Ajax: A new approach to web applications, February 2005. URL <http://adaptivepath.com/ideas/essays/archives/000385.php>.
- [36] Delphi Group. The value of standards. Survey, Delphi Group, Ten Post Office Square, Boston, MA 02109, June 2003. www.ec-gis.org/sdi//ws/costbenefit2006/reference/20030728-standards.pdf.
- [37] Martin Gudgin, Yves Lafon, and Anish Karmarkar. Resource representation SOAP header block. W3C recommendation, W3C, January 2005. <http://www.w3.org/TR/2005/REC-soap12-rep-20050125/>.
- [38] Martin Gudgin, Martin Gudgin, Marc Hadley, Tony Rogers, Tony Rogers, and Marc Hadley. Web services addressing 1.0 - SOAP binding. W3C recommendation, W3C, May 2006. <http://www.w3.org/TR/2006/REC-ws-addr-soap-20060509>.
- [39] Martin Gudgin, Marc Hadley, and Tony Rogers. Web services addressing 1.0 - core. W3C recommendation, W3C, May 2006. <http://www.w3.org/TR/2006/REC-ws-addr-core-20060509>.
- [40] Hugo Haas, David Booth, Eric Newcomer, Mike Champion, David Orchard, Christopher Ferris, and Francis McCabe. Web services architecture. W3C note, W3C, February 2004. <http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/>.
- [41] Red Hat. Hibernate Reference Documentation 3.3.1. Technical report, September 2008. http://www.hibernate.org/hib_docs/v3/reference/en-US/pdf/hibernate_reference.pdf.
- [42] Hi-G-Tek. URL <http://www.higtek.com/>.
- [43] David Hyatt and Ian Hickson. HTML 5. W3C working draft, W3C, February 2009. <http://www.w3.org/TR/2009/WD-html5-20090212/>.
- [44] Melissa Irmen. 10 ways to reduce the cost and risk of global trade management. *Journal of Commerce*, March 2009. <http://www.joc.com/node/410216>.
- [45] Martin Kalin. *Java Web Services: Up and Running*. O'Reilly, February 2009.
- [46] Michael Kay. XSL transformations (XSLT) version 2.0. W3C recommendation, W3C, January 2007. <http://www.w3.org/TR/2007/REC-xslt20-20070123/>.
- [47] Christian Kiehle, Klaus Greve, and Christian Heier. Requirements for next generation spatial data infrastructures-standardized web based geoprocessing and web service orchestration. *Transactions in GIS*, 11(6):p819 – 834, 20071201. ISSN

13611682. URL <http://search.ebscohost.com.www2.lib.ku.edu:2048/login.aspx?direct=true&db=aph&AN=28048260&site=ehost-live>.
- [48] Yves Lafon and Nilo Mitra. SOAP version 1.2 part 0: Primer (second edition). W3C recommendation, W3C, April 2007. <http://www.w3.org/TR/2007/REC-soap12-part0-20070427/>.
- [49] Kelvin Lawrence, Chris Kaler, Anthony Nadalin, Ronald Monzillo, and Phillip Hallam-Baker. Web Services Security: SOAP Message Security 1.1 (WS-Security 2004). OASIS standard, OASIS, February 2006. <http://docs.oasis-open.org/wss/v1.1/wss-v1.1-spec-os-SOAPMessageSecurity.pdf>.
- [50] Kelvin Lawrence, Chris Kaler, Anthony Nadalin, Marc Goodner, Martin Gudgin, Abbie Barbir, and Hans Granqvist. WS-SecurityPolicy 1.2. OASIS standard, OASIS, July 2007. <http://docs.oasis-open.org/ws-sx/ws-securitypolicy/v1.2/ws-securitypolicy.pdf>.
- [51] E. Levinson. The MIME Multipart/Related Content-type. RFC 2387 (Proposed Standard), August 1998. URL <http://www.ietf.org/rfc/rfc2387.txt>.
- [52] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler. Tinyos: An operating system for sensor networks. In *in Ambient Intelligence*. Springer Verlag, 2004. <http://www.cs.berkeley.edu/~culler/AIIT/papers/TinyOS/levis06tinyos.pdf>.
- [53] Amelia A. Lewis. Web services description language (WSDL) version 2.0: Additional MEPs. W3C note, W3C, June 2007. <http://www.w3.org/TR/2007/NOTE-wsdl20-additional-meps-20070626>.
- [54] Canyang Kevin Liu. First Look at WSDL 2.0. SAP article, SAP, January 2005. <https://www.sdn.sap.com/irj/scn/go/portal/prtroot/docs/library/uuid/74bae690-0201-0010-71a5-9da49f4a53e2>.
- [55] C. Matthew MacKenzie, Ken Laskey, Francis McCabe, Peter F Brown, Rebekah Metz, and Booz Allen Hamilton. Reference Model for Service Oriented Architecture 1.0. OASIS standard, OASIS, October 2006. <http://docs.oasis-open.org/soa-rm/v1.0/>.
- [56] Lance McKee. The Importance of Going “Open”. OGC white paper, OGC, July 2005. http://portal.opengeospatial.org/files/?artifact_id=6211.
- [57] Noah Mendelsohn, Murray Maloney, Henry S. Thompson, and David Beech. XML schema part 1: Structures second edition. W3C recommendation, W3C, October 2004. <http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/>.
- [58] Noah Mendelsohn, Hervé Ruellan, Martin Gudgin, and Mark Nottingham. XML-binary optimized packaging. W3C recommendation, W3C, January 2005. <http://www.w3.org/TR/2005/REC-xop10-20050125/>.

- [59] Jean-Jacques Moreau, Sanjiva Weerawarana, Roberto Chinnici, and Arthur Ryman. Web services description language (WSDL) version 2.0 part 1: Core language. W3C recommendation, W3C, June 2007. <http://www.w3.org/TR/2007/REC-wsd120-20070626>.
- [60] Arthur Na and Mark Priest. Sensor Observation Service. OGC implementation specification, OGC, October 2007. http://portal.opengeospatial.org/files/?artifact_id=26667.
- [61] Arthur Na and Mark Priest. Sensor Observation Service. OGC schema, OGC. <http://schemas.opengis.net/sos/>.
- [62] Douglas Nebert, Arliss Whiteside, and Panagiotis (Peter) Vretanos. OpenGIS Catalogue Services Schemas. OGC schema, OGC. <http://schemas.opengis.net/csw/>.
- [63] Douglas Nebert, Arliss Whiteside, and Panagiotis (Peter) Vretanos. OpenGIS Catalogue Services Specification. OGC implementation specification, OGC, February 2007. http://portal.opengeospatial.org/files/?artifact_id=20555.
- [64] Eric Newcomer and Greg Lomow. *Understanding SOA with Web Services (Independent Technology Guides)*. Addison-Wesley Professional, December 2004. ISBN 0-321-18086-0. <http://portal.acm.org/citation.cfm?id=1044935>.
- [65] Duane Nickul, Laurel Reitman, James Ward, and Jack Wilber. Service Oriented Architecture (SOA) and Specialized Messaging Patterns. Adobe article, Adobe, December 2007. www.adobe.com/enterprise/pdfs/Services_Oriented_Architecture_from_Adobe.pdf.
- [66] Henrik Frystyk Nielsen, Marc Hadley, Anish Karmarkar, Noah Mendelsohn, Yves Lafon, Martin Gudgin, and Jean-Jacques Moreau. SOAP version 1.2 part 1: Messaging framework (second edition). W3C recommendation, W3C, April 2007. <http://www.w3.org/TR/2007/REC-soap12-part1-20070427/>.
- [67] Henrik Frystyk Nielsen, Anish Karmarkar, Noah Mendelsohn, Martin Gudgin, Yves Lafon, Marc Hadley, and Jean-Jacques Moreau. SOAP version 1.2 part 2: Adjuncts (second edition). W3C recommendation, W3C, April 2007. <http://www.w3.org/TR/2007/REC-soap12-part2-20070427/>.
- [68] Mark Nottingham, Hervé Ruellan, Noah Mendelsohn, and Martin Gudgin. SOAP message transmission optimization mechanism. W3C recommendation, W3C, January 2005. <http://www.w3.org/TR/2005/REC-soap12-mtom-20050125/>.
- [69] David Orchard, Hugo Haas, Sanjiva Weerawarana, Amelia A. Lewis, Roberto Chinnici, and Jean-Jacques Moreau. Web services description language (WSDL) version 2.0 part 2: Adjuncts. W3C recommendation, W3C, June 2007. <http://www.w3.org/TR/2007/REC-wsd120-adjuncts-20070626>.

- [70] Cesare Pautasso, Olaf Zimmermann, and Frank Leymann. Restful web services vs. "big" web services: making the right architectural decision. In *WWW '08: Proceeding of the 17th international conference on World Wide Web*, pages 805–814, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-085-2. URL <http://doi.acm.org/10.1145/1367497.1367606>.
- [71] George Percivall, Carl Reed, Lew Leinenweber, Chris Tucker, and Tina Cary. OGC Reference Model. Technical report, OGC, November 2008. http://portal.opengeospatial.org/files/?artifact_id=31112.
- [72] Clemens Portele. OpenGIS Geography Markup Language (GML) Encoding Standard. OGC implementation specification, OGC, August 2007. http://portal.opengeospatial.org/files/?artifact_id=20509.
- [73] Clemens Portele. OpenGIS Geography Markup Language (GML) Encoding Standard. OGC schema, OGC. <http://schemas.opengis.net/gml/>.
- [74] Carl Reed. Topic 0: Abstract Specification Overview. OGC abstract specification, OGC, June 2005. http://portal.opengeospatial.org/files/?artifact_id=7560.
- [75] Mark Reichardt. The Havoc of Non-Interoperability. OGC white paper, OGC, December 2004. http://portal.opengeospatial.org/files/?artifact_id=5097.
- [76] Leon S. Searl. Service Oriented Architecture for Sensor Networks Based on the Ambient Computing Environment. ITTC technical report, ITTC, February 2008. www.ittc.ku.edu/sensornet/trusted_cooridors/papers/41420-07.pdf.
- [77] Jérôme Siméon, Don Chamberlin, Daniela Florescu, Scott Boag, Mary F. Fernández, and Jonathan Robie. XQuery 1.0: An XML query language. W3C recommendation, W3C, January 2007. <http://www.w3.org/TR/2007/REC-xquery-20070123/>.
- [78] Ingo Simonis and Johannes Echterhoff. Sensor Alert Service. OGC candidate implementation specification, OGC, June 2007. http://portal.opengeospatial.org/files/?artifact_id=24780.
- [79] KC SmartPort. Trade Data Exchange - Nothing short of a logistics revolution. *Journal of Commerce*, November 2008. URL <http://www.joc-digital.com/joc/20081110/?pg=29>.
- [80] Dennis Sosnoski. JiXB, March 2009. URL <http://jibx.sourceforge.net/>.
- [81] C. M. Sperberg-McQueen, François Yergeau, Eve Maler, Jean Paoli, and Tim Bray. Extensible markup language (XML) 1.0 (fifth edition). W3C proposed edited recommendation, W3C, February 2008. <http://www.w3.org/TR/2008/PER-xml-20080205>.

- [82] Anne van Kesteren. HTML 5 differences from HTML 4. W3C working draft, W3C, June 2008. <http://www.w3.org/TR/2008/WD-html5-diff-20080610/>.
- [83] Anne van Kesteren. The XMLHttpRequest object. a WD in last call, W3C, April 2008. <http://www.w3.org/TR/2008/WD-XMLHttpRequest-20080415/>.
- [84] Anne van Kesteren and Ian Hickson. Offline web applications. W3C note, W3C, May 2008. <http://www.w3.org/TR/2008/NOTE-offline-webapps-20080530/>.
- [85] Michael Wolfe. In this case, bad news is good news. *Journal of Commerce*, July 2004. www.ismasecurity.com/ewcommon/tools/download.aspx?docId=175.



Technical Report

Application of Passive UHF RFID in Intermodal Facilities

Daniel D. Deavours

ITTC-FY2010-TR-41420-14

July 2009

Project Sponsor:
Oak Ridge National Laboratory

Abstract

In this paper, we explore the use of passive UHF RFID for tracking containers within intermodal facilities. We find that there are a number of emerging technologies that may be applicable, including the relatively new Mojix system and the KU Cargo Tag, a long range passive UHF RFID tag developed at the University of Kansas. We note that while there may be technical solutions available, the more challenging issue involves coordination and cooperation with the large number of companies and agencies that would benefit from such a system.

Table of Contents

Abstract.....	i
Table of Contents.....	ii
List of Figures.....	ii
1. Introduction.....	1
2. Background.....	2
2.1. Passive UHF RFID	2
2.2. KU Cargo Tag.....	3
2.3. Portunus RFID Tag.....	4
2.4. Mojix.....	5
2.5 RF Controls.....	6
2.6. Advanced Research in Location Positioning in Multi-Path Environments	8
3. Observations	9
3.1. Manufacturing and Execution System (MES).....	9
3.2. Dock Door Application.....	11
3.3. Real Time Location Positioning (RTLS).....	11
3.4. Large Container Stationing	12
4. Applicability to Intermodal Facilities	13
5. Conclusions.....	15
References.....	16

List of Figures

Figure 1: A popular commercial passive UHF RFID tag.....	2
Figure 2: The KU Cargo tag.	4
Figure 3: Early version of the Portunus RFID tag. Photo by Megan E. Gannon.....	5
Figure 4: Mojix Star node (a) and 4-port eNode (b) [9].	6
Figure 5: RF Controls unit.	7
Figure 6: Experimental MES line.	10
Figure 7: Example of experimental test scenario for large metal container storage and staging area.....	13

1. Introduction

Automatic container-tracking within an intermodal facility may improve logistical efficiency and security. Passive UHF RFID is a potentially useful technology for tracking containers because of a unique combination of attributes, such as cost, performance, standardization, and adoption. An emerging, new capability within passive UHF RFID tag community is the ability to accurately identify the positioning of tags. This document reviews our findings on the suitability of using passive UHF RFID for intermodal facilities, and in particular on the applicability of location positioning capabilities within such an environment.

Recently, there has been interest in the development of passive UHF RFID (pRFID) tags for intermodal containers. The rationales for using pRFID for intermodal containers are briefly as follows.

- Relatively long read distance allows automatic identification at range (30 feet or more).
- Relatively low cost (few cents to dollars US).
- The lack of an internal power supply reduces maintenance and extends life of the product to approximately that of the life of the asset.
- Compatible with ISO 18000-6c (“Gen 2”), which is becoming a very well-supported standard.
- Compatible with related EPCglobal information-sharing standards to associate containers with pallets, cases, and items within those containers and to share that information across organizational boundaries.
- Weak but functional authentication through a combination of techniques, which gives reasonably strong pedigree verification. Techniques include:
 - Use of unique tag ID (TID) within each IC that requires a foundry to replicate.
 - Password-protected portions of memory

Because of these attributes, the use of pRFID may be a suitable technology for automatic tracking of containers.

Some challenges of using the technology, briefly, are as follows.

- Typically relatively short read distance (only 30 feet or so) as compared to active RFID
- Little intelligence / sensing / positioning
- Lack of existing products for cargo containers
- Lack of standardization, positive pilots, and proven business cases

While the above list is not exhaustive, they are probably the most notable. In this report, we aim to report on the current state of the technology to address the technical challenges, and we give comments on related business and policy challenges.

The remainder of the report is outlined as follows. In Section 2, we give a further review of the pRFID technology to date, including ITTC/KU contributions, and emerging commercial products that provide location-positioning information. In Section 3, we give an overview of our observations of the technology in action. It was our aim to observe the technology used in use cases as similar as possible to those of an intermodal facility, but due to the newness of the

technology and other constraints, that was not possible. However, we give a report on the technology's capabilities as we observed them in the fall of 2008. In Section 4, we extrapolate our observations and knowledge of the pRFID system to pRFID may be used in an intermodal facility. Finally, we conclude with observations and thoughts on future directions.

2. Background

2.1. Passive UHF RFID

Passive RFID is an old technology that has likely been used since WW-II [1]. Passive UHF RFID (hereafter pRFID), or RFID that specifically uses the frequency band between 860 and 960 MHz, has recently gained attention primarily because of retailer mandates [2], in which retailers are attempting to tag cases and pallets of products moving through the supply chain. pRFID transponders, or "tags," are interrogated by interrogators or "readers." The reader RF energy is used to power the tag (hence the tags are passive), interact with the tag according to a protocol, provide a clock, and to provide a carrier signal for the tag to backscatter information. The tags contain an integrated circuit (IC) which is used to decode the reader signal, store the tag ID in non-volatile RAM, and to backscatter the ID and other information. The tag also consists of an antenna, some carrier or encapsulation, and a method of attachment, such as pressure sensitive adhesive. The tag backscatters a signal by changing the impedance of the IC, which can change the magnitude and phase of the signal scattered (retransmitted) by the antenna.

The hope within the supply chain industry is that increased product visibility will result in increases in efficiencies, such as reduced out-of-stocks, better logistical planning, anti-counterfeit measure, and reduced labor costs. UHF RFID was chosen because of a combination of low-cost (\$0.10 per tag is now common) and relatively long read distance (25 feet is common), which will facilitate reading cases on a conveyor as well as cases and pallets moving through a dock door. Figure 1 shows a popular pRFID tag roughly to scale.



Figure 1: A popular commercial passive UHF RFID tag.

The demand for pRFID for has driven a number of processes and products. Most importantly, there was a concerted effort within the community to develop a set of standards around the use of pRFID. These include the air interface protocol (ISO 18000-6c [3]), standard ways to interact with readers, and ways to store and share information within an organization, as well as between organizations. These standards have been managed by the EPCglobal, a standards body that promotes Electronic Product Code through RFID. The standardization, as well as the demand, has resulted in high quality, low cost sources for tags, readers, and information systems. This is a potentially rich infrastructure for intermodal transport tracking to build upon. Part of that infrastructure is the development of RFID tags that are designed for use on cargo containers that can be read at long distances.

2.2. KU Cargo Tag

The KU Cargo tag has been described in [4]. The KU Cargo tag is intended to be affixed to large metal assets and to be read from a long distance. A prototype was constructed out of a six inch square polypropylene plastic sheet that is approximately 5 mm thick. On the reverse side we attach a copper foil, which acts as a ground plane when the tag is not attached to a metal surface. On the front side we attach an “inlay”, which consists of an antenna and an IC. The inlay size is approximately 4 inches square. The antenna is designed to provide excellent efficiency (less than 1 dB loss), relatively wide bandwidth, and large antenna gain. The measured gain of the tag antenna is over 5 dBi.

Briefly, the KU Cargo tag is a microstrip antenna with the TM_{01} and TM_{10} modes placed at 867 and 915 MHz so that it can operate efficiently both within the 865-869 MHz frequency band (used in Europe and many other countries) and 902-928 MHz (used in North America and many Asian countries). In one band, the tag is horizontally polarized, and in a second band, it is vertically polarized. As with dipole antennas, this usually imposes a 3 dB polarization loss when used with circularly polarized reader antennas, but since orientation on containers can be controlled, use of linearly polarized reader antennas is likely possible for further improved efficiency. Unlike dipole antennas, the Cargo tag provides a high gain over a wide bandwidth when attached to large containers, and thus can be read at considerably longer distance.

What is novel about the KU Cargo tag is that we inscribe the antenna with a cross-shaped slot, and we place a matching circuit within that slot. We feed the two modes in series. Conveniently, when one mode is active, the other mode presents a low impedance (close to that of a short circuit). Using that, we are able to achieve an excellent impedance match over both frequency bands.

Finally, since the writing of [4], we have modified the matching circuit to accept the Alien Higgs 3 IC [5]. The original IC had a minimum turn-on power of -13 dBm, while the Higgs 3 IC has a stated turn-on power of approximately -18 dBm, which yields more than 5 dB of performance improvement over [4].



Figure 2: The KU Cargo tag.

The KU Cargo tag designed for the Higgs 3 IC, shown in Figure 2, was tested by a third party in a warehouse environment with a circularly polarized reader antenna. Note that this environment provided considerable ground, ceiling, and other sources of reflection. We observed reliable reads out to approximately 80 feet and intermittent reads to about 100 feet. From about 110 to 120 feet, the tag was not readable likely because it was in a local null zone. At about 130 feet, the tag became readable intermittently. The long read distance was likely assisted by multi-path effects in the warehouse environment.

2.3. Portunus RFID Tag

The Portunus RFID tag [6] (see Figure 3) was designed to operate in a heavy industrial environment. The requirement was to have a low profile, be very rugged, and it was acceptable

to compromise read distance for other requirements. The Portunus tag was developed by the University of Kansas RFID Lab to meet this market requirement. The Portunus tag is two by four inches square and uses a 1/16" polycarbonate substrate. The IC is protected by being placed in a small recess in the polycarbonate. Again, the inlay is affixed to the front of the substrate and a copper foil to the reverse side of the substrate. More information about the technology used to develop the RFID tag is given in [7].

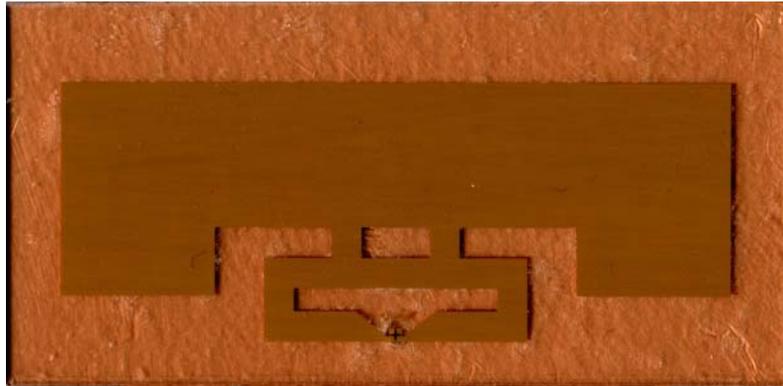


Figure 3: Early version of the Portunus RFID tag. Photo by Megan E. Gannon.

The Portunus RFID tag is currently being used to tag metal assets for use in rugged environments, such as manufacturing. The largest application is installation on farm equipment at the time of manufacturing. Since the Portunus tag is commercially available, it was used for some of the location-positioning testing.

2.4. Mojix

In traditional pRFID systems, the readers emit RF energy fields, and tags within the field respond. The reader performs the transmitting and receiving functions, i.e., a transceiver, either by using two isolated antennas (bistatic), or electrically splitting the transmit and receive signals from a single antenna (monostatic). Readers are relatively self-contained, typically having a considerable amount of local computing resources to run local applications, but also capable of communicating to middleware or back-end systems through a network. To instrument a large facility with RFID, it takes a large number of readers, each covering a relatively small localized zone. While some readers are becoming smarter and capable of some positioning or sensing capabilities, such as a direction of travel through a portal or sensing a tag is getting closer or further away, location positioning is dominated by presence or absence in read zones and travel through choke points. These capabilities are of little use in intermodal facilities.

The Mojix system [8] differs from the classical system in a number of ways. First, the hardware performing transmit and receive function are physically separated. The transmit function is performed a low-cost "eNode" [9] (see Figure 4b). The receive function over a large area is concentrated in a single (or small number) of "Star" nodes [10] (see Figure 4a), which contain a phased array antenna and sophisticated signal processing capabilities. The Star node controls the eNodes through a coaxial cable or wirelessly, and provides power, commands, and precise signaling information. The Star system also uses a phased array, very sensitive receivers, and sophisticated signal processing to detect tags that have been excited by eNodes at very long

distances (200 meters or more). Using a combination of techniques, the Star is able to provide reasonably accurate location-positioning information. Product literature indicates a positioning accuracy of one meter or better.

The complete operating process of the Mojix system is proprietary. We speculate that the following techniques are used in combination to determine the position of an RFID tag.

- The set of eNodes that is able to excite a tag. The tag lies in the intersection of the read fields of each of the eNodes.
- Varying the power of the eNode until a tag is no longer in field. The tag location can be localized to the surface of a sphere.
- Tag signal strength. Tags that are close to eNodes receive more power and (presumably) have a larger backscatter signal. By measuring the quality of the tag signal, one can estimate the distance of the tag from the eNode.
- Angle of arrival. Using array processing and independent, synchronized receivers on each element of the antenna array, the phase angle of the received signal at each antenna in the element is likely to be estimated with good accuracy.

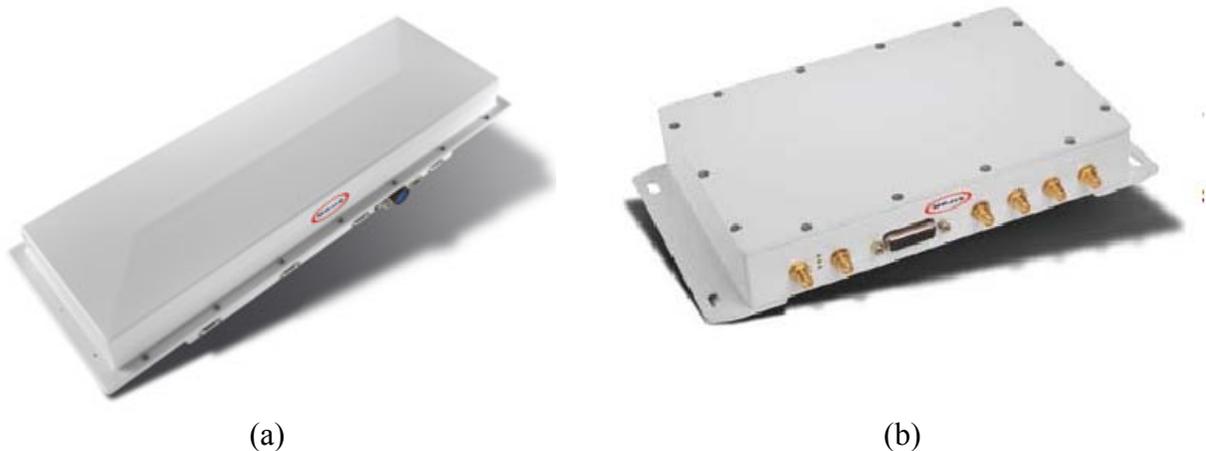


Figure 4: Mojix Star node (a) and 4-port eNode (b) [9].

It is also possible that the Mojix system uses techniques that are discussed in Section 2.6.

2.5 RF Controls

RF Controls [11] produces a product that has similarities to that of Mojix in that it provides a level of location-positioning information. The architecture of the RF Controls system is what they call the “Integrated Antenna Array”, or IAA. Figure 5 is an illustration of an IAA with the back cover removed. The IAA consists of a commodity RFID reader, some unknown hardware, and approximately 32 antenna elements. The IAA works as a single, large phased array for monostatic operation. We observed a 4-way power splitter / combiner on the back of the array, which is likely used to split the signal from / to the RFID reader to / from the array elements. Thus, we surmise that the 32 elements are divided into four blocks of 8 elements, with each block of 8 receiving equal power. Though hidden, we suspect that there is a phase shifter on

each of the array elements. By controlling the phase of each of the elements, one is able to create a highly directional antenna.

We were able to observe a single IAA system in operation in a limited field trial. The system was not fully configured when we observed operation. However, we were able to observe a demonstration in which showed a tag position being correctly identified to within approximately one meter of accuracy from a distance of approximately 5 meters. Again, it is difficult to conclude much from this limited observation.

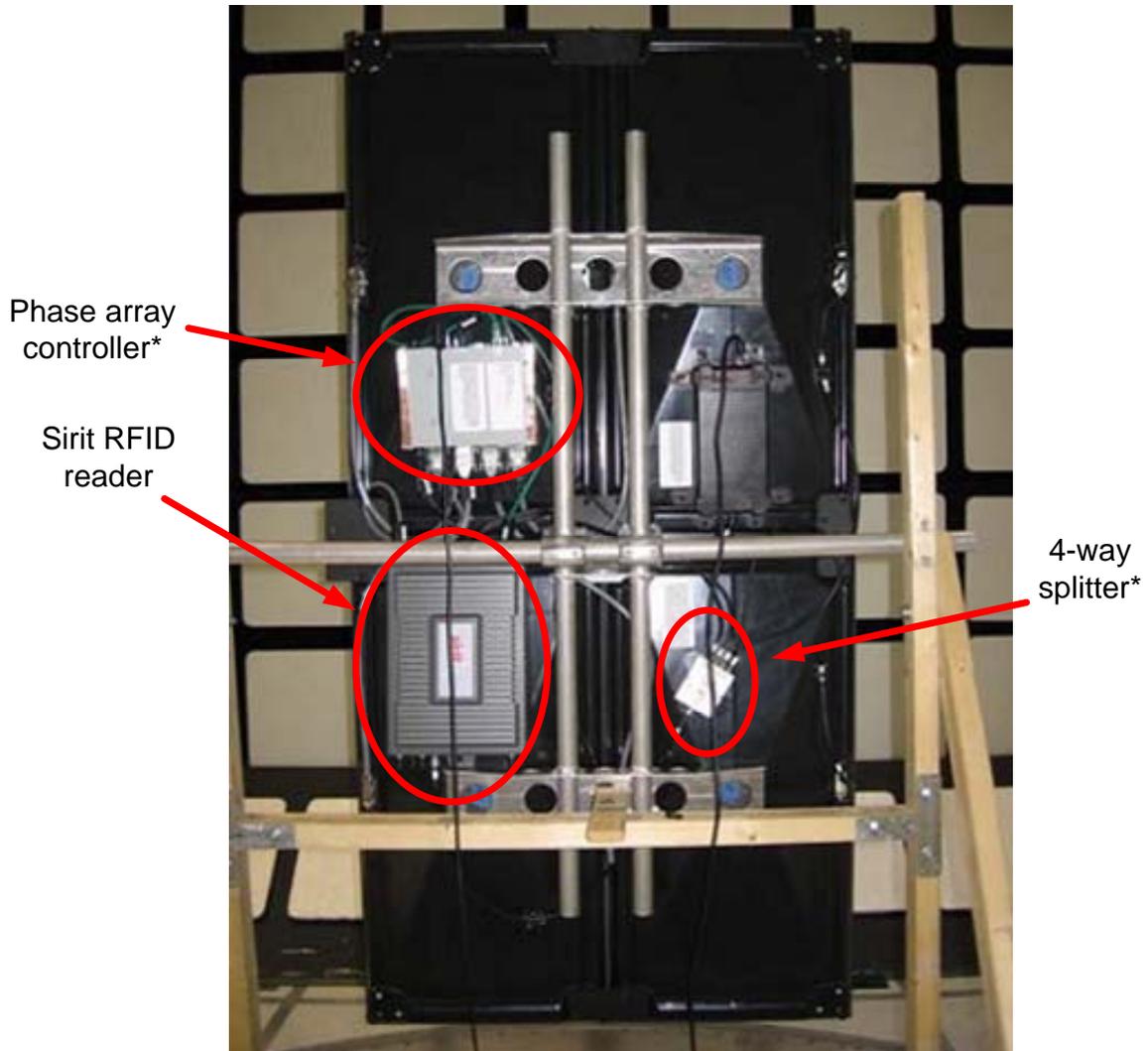


Figure 5: RF Controls unit.

From what we know of the architecture, we can conclude several things.

- Because the IAA is built from commodity components, the initial and overall system cost is likely to be low.
- Because the system is inherently monostatic, one complete IAA system will need to be installed for each area to be monitored.

- By overlapping read zones, one is able to increase the positioning accuracy of the system, but with a proportionate increase in cost.
- A RF Controls representative indicated that positioning was likely to be inaccurate at distance and in heavy multi-path environments.
- The phased array is physically large, approximately 4 by 8 feet. This makes it impractical for many outdoor environments where wind load will be an issue.

The economics of the RF Controls and Mojix system differs. The IAA is likely to have a lower initial cost, but will scale more poorly. The Mojix system, because of the cost of the Star receiver, will have a high initial cost. Adding additional eNodes are likely to be much less expensive. Neither RF Controls nor Mojix were willing to provide any pricing information outside of a confidentiality agreement.

2.6. Advanced Research in Location Positioning in Multi-Path Environments

A recent article [12] described some of the most advanced research in location positioning in a multi-path environment. We briefly describe the approach that we investigated here.

The premise of this approach is that the receiver uses a phased array that is very well characterized and calibrated. Second, it is assumed that the signal from the transmitter has a limited (20 kHz) bandwidth, which is comparable to signal bandwidth used in RFID systems. The system uses a combination of both the estimated angle of arrival and the signal correlation coefficient to estimate whether two signals that are received from different angles are in fact the same signal.

Within an RFID system, the question is not whether the signal is emitted from two different sources or the same source; a collision detection algorithm within the protocol can be used for that purpose. The difference in a Mojix-like system is that the system is coherent: the tag's signal is the modulated reflected signal from an eNode, which is under direct and presumably coherent control from the Star node. Thus, it may be possible to use a coherent match filter and the angle of arrival to reinforce the estimate of a distinct multi-path. Further, the matched filter can be used to estimate the path length of each incident signal.

If one is able to correctly estimate all the multi-paths and the length of each path, then one may be able to estimate the distance of the shortest path. The distance and angle of arrival of the shortest distance, if line-of-sight is one of the paths, is sufficient to estimate the location of a tag. Failing that, one may be able to estimate the angle and distance of the dominant signal. Again, the estimate would assume the dominant signal is the line-of-sight signal, and that a multi-path signal is attenuated.

As with any FCC-compliant device, the RFID reader system uses frequency hopping over the 902-928 MHz frequency band, so tags may be interrogated multiple times on different channels in order to obtain a more reliable estimate. Finally, for stationary or slow-moving objects, repeated observations may be used to increase the confidence of various estimates.

We emphasize that we have no knowledge about whether the Mojix system uses this or any similar approach to estimating positioning. We are merely inferring that existing research results of [12] with modifications may be applied to a Mojix-like system for local positioning.

3. Observations

Here, we report on our observations of the Mojix system used for evaluation of commercial fitness for a number of use cases. We are somewhat limited in what we are able to report due to sensitivity of the companies involved.

3.1. Manufacturing and Execution System (MES)

We observed Mojix operational in a MES (manufacturing and execution system, i.e., an assembly line) use case. Figure 6 illustrates the MES setup that was tested. We observed two use cases. First, a metal rack on wheels was tagged with a KU Portunus RFID tag. The metal rack was pulled through five stations. At each station, the rack was placed in three positions: early, middle, and late within the station, labeled A, B, and C respectively. At each point in the station, the Mojix system attempted to identify the station that the rack was in, and in which position of each station. The experiment was repeated five times. We observed that for all five experiments, the Mojix system was able to correctly identify the station and position within the station with every attempt.

The second use case involved two metal racks, both tagged using a KU Portunus tag. The two racks were exactly one station apart. For example, the first rack was in 1A and the second rack was in 2A, then the first rack was moved to 1B and the second rack to 2B, etc. The experiment was repeated five times. Again, the Mojix system was able to correctly identify the station and position of each rack in all five of the experiments.

We speculate that the method in which the Mojix system is estimating the rack position is by determining the eNodes that are able to excite the Portunus tags and the signal strength of each of the tags as excited by different eNodes, and perhaps by varying the eNode transmit power. For example, a rack at position 1B could be read by eNode 1 on very low power levels. In position 1C, eNode 1 would require more power to read the tag, while eNode 2 may be able to read the tag at very high power levels.

Star Node

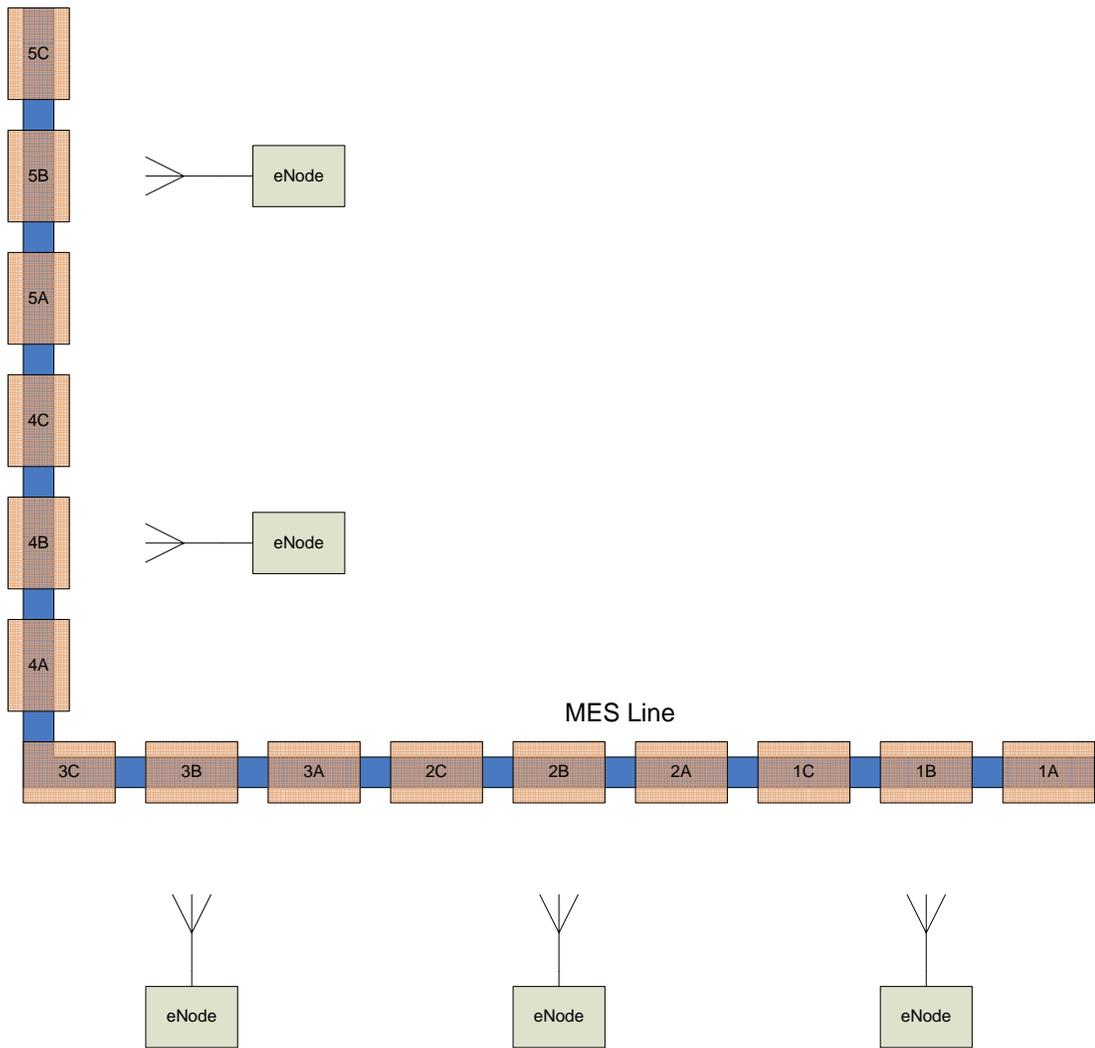


Figure 6: Experimental MES line.

3.2. Dock Door Application

Next, the Mojix system was used to identify pallets coming in and out of a dock door onto a truck. Two dock doors were instrumented with eNodes and photo sensors. The test was to determine if the Mojix system could correctly identify the pallet tag ID and the door that the pallet would enter or exit while being carried by a fork lift. The test setup was less formal, and was performed relatively quickly based on time constraints. We observed that the Mojix system was able to correctly identify the dock door and the pallet tag on every test case.

This test seems somewhat contrived, since there was only one lift truck operational, and there were photo sensors on both of the eNodes. Simply correlating the reads with the photo sensor would be sufficient for the system to correctly correlate the reads with the dock door. Thus, this test demonstrated no unique capability of the Mojix system.

In private discussions with Mojix representatives, Mojix claims that their system is capable of detecting the contents of pallets on fork lifts traveling through as many as eight dock doors simultaneously. We have no way of verifying this claim, nor do we have any reason to doubt its validity.

This form of location-positioning offers little technical advantage over the capability of current fixed reader systems, but it was a successful demonstration.

3.3. Real Time Location Positioning (RTLS)

Next, we observed the Mojix system acting in a RTLS tracking mode in tracking a single RFID tag. This proved problematic. The team spent two days troubleshooting the system, and the system never fully functioned properly. Debriefing with Mojix representatives, we understand that there were some configuration difficulties and challenges working across time zones, and that Mojix did not understand that this would be a tested use case and therefore the system was not configured to perform that test. Regardless, here are our general observations.

- The RTLS system took several seconds to identify the region and provide an $\langle X, Y \rangle$ coordinate for the system. Once it did, the system appeared to be fairly accurate, i.e., within three to five feet, sometimes more accurate.
- If the tag moved, the RTLS system would take several seconds to recognize the move, and the measured position would slowly converge to the new position over several more seconds. After debriefing with Mojix, we understand that this was an intentional delay filter that was inserted because of the previous tests involving tracking racks required accuracy of slow-moving objects.
- The RTLS did not appear to be able to accurately track a tag moving at the pace of a brisk walk, or any motion that involved several abrupt changes in direction. After debriefing with Mojix, it is their claim that the system is capable of providing that capability, but the test that we observed was not properly configured for that use case. We have no evidence to confirm or reason to refute this claim.

3.4. Large Container Stationing

This example is very similar to the use of ISO containers in an intermodal facility. In this example, the KU Cargo tag was attached to a large metal container. While these were not ISO containers, there were similarly sized and used in a similar manner.

The particular application studied here is to manage large containers in a staging yard. The yard is 150 feet wide and contains numerous containers that are moved by heavy equipment. Within this 150 zone, no RFID equipment is allowed. The application is to track the container assets within the yard. Assets may be at most 75 feet from one of the sides of the yard, and thus there is a tag read distance requirement of 75 feet. In this experiment, the KU Cargo tag using the Higgs 3 IC was affixed to a container and the Mojix system was used to track the location of the container within the yard. The KU Cargo / Higgs 3 tag was used because of the long read distance requirement.

To maximize performance, the Mojix eNodes should use linearly polarized antennas, since the KU Cargo tag is linearly polarized. This experiment was found to be satisfactory using circularly polarized eNode antennas. The Mojix system was deployed around the perimeter of the yard to provide location-positioning information. Figure 7 illustrates the setup.

We are able to give only high-level observations of the results. We can say that the Mojix system was able to reasonably predict the location of RFID tags within the staging area and with reasonable accuracy and timeliness. We generally conclude that with line-of-sight, the Mojix system is likely to be able to detect with reasonable accuracy the position of a container in a yard.

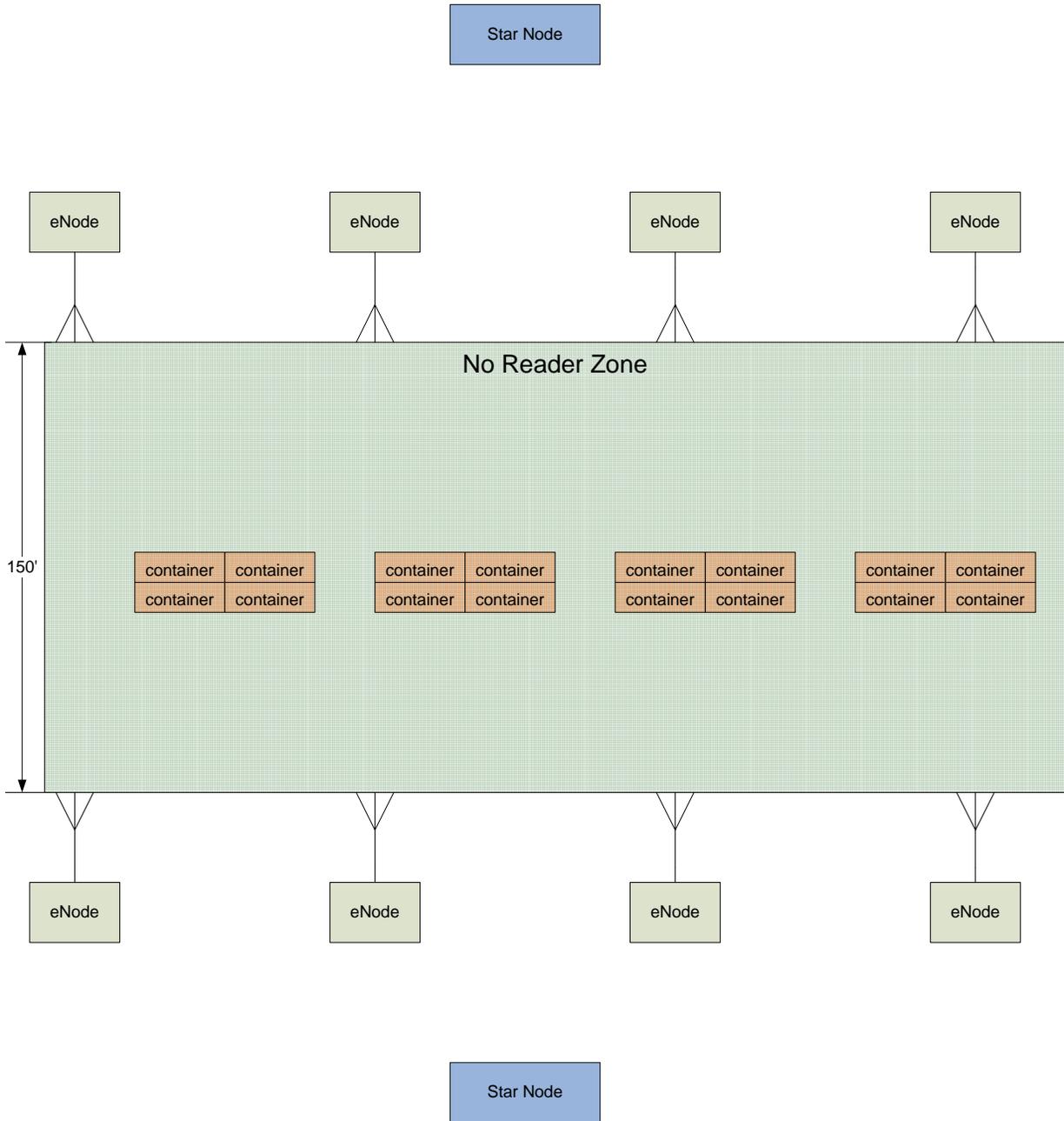


Figure 7: Example of experimental test scenario for large metal container storage and staging area.

4. Applicability to Intermodal Facilities

We were not able to directly test the ability to tag and track containers in intermodal facilities. This is due to relatively new technology and great sensitivity from potential early adopters to protect potential trade secrets. However, we have seen the product perform in several environments, have interacted with representatives, and can discuss potential capabilities.

We note the following observations:

- As claimed [13], the Star node is extremely sensitive. It was able to detect the KU Cargo tag at nearly 130 feet away when the Star was pointing away from the tag. Thus, the signal from the tag was very weak, and the Star node must have detected the signal from one or more reflections, further weakening the signal.
- We observed location-positioning capabilities that were not highly refined and took some time to settle on a location, but likely due to in part to improper configuration. Regardless, it showed location positioning accuracy sufficient to one to two meters, which is more than sufficient for locating a container in a storage yard. Similarly positive results from the container staging area are positive indicators of likely usability.
- The wireless eNode released in October 2008, which we did not observe, potentially adds flexibility to the excitors. It is not clear whether the wireless eNodes can be mobile and still provides accurate location positioning, or whether it is a feature that could be added in the future. If so, a mobile eNode could be used inventory and position containers within the yard periodically.
- The KU Cargo tag provided excellent long-distance performance. IC improvement has been growing steadily so that read distances double every 2 to 3 years. There is evidence that this trend can continue for at least one more generation of chips before efficiencies are exhausted and performance becomes more closely tied to Moore's law. However, recent trends indicate that additional improvements will be focused on decreasing costs more than increasing IC sensitivity, so it might be the case that the next generation of ICs will provide more features and lower cost rather than improved performance.

Within an intermodal facility, we envision two possible use cases. The first is to observe the container any time it is moved, and the second is to observe the container as it is stationary. Both provide different use cases and potential business benefits.

All loading and moving equipment would be fitted with a powered eNode, which would be able to excite RFID tags attached to containers as it is being moved. All truck and rail chokepoints would also be fitted with powered eNodes and sensors. One or two Star nodes would be fitted within a facility that would provide coverage and greater accuracy in triangulation. In this way, all movement of containers within the facility could be directly monitored with a system like the Mojix RTLS and supplemented through other forms of data acquisition, such as manual data entry. With the wireless eNode, these use cases become readily feasible.

Static monitoring is useful to detect unauthorized or erroneous movement of containers within the facility. This can be very helpful, for example, in finding "lost" containers. For this application, one could fit normal operating equipment with a wireless eNode, which would move throughout the yard in normal operations, and in doing so, would assist in monitoring location information of the containers in the yard. In addition, a periodic yard "audit" could be performed by one of the vehicles to validate the yard inventory. These methods would allow the excitation of RFID tags potentially deep within container "canyons." Without direct line-of-sight, location-positioning is not likely to be highly accurate. However, accuracy of this use case is not critical; it would be sufficiently accurate to detect whether a container is not close to where it is supposed to be, or is absent. Anomalous container movement could be detected relatively quickly, and still protected by chokepoints.

While use of RFID and location services is useful for yard management, more value can be obtained through better logistics management. Knowing when and where containers were offloaded could provide better end-to-end logistics management. For example, if a container is not off-loaded at the proper port (perhaps because it was improperly loaded), the sooner the information is known, the easier it is for logistics providers and down-stream consumers to make adjustments.

Second, by using the authentication features of the ISO 18000-6c standard, one can perform better and more targeted container inspections. By having reasonably strong authentication and methods of sharing data, together with manifest information, one can easily identify containers that come from trusted sources through secure ports and those that do not. This will allow port security to target those containers that are at much higher risk.

It is our opinion that pRFID will not be implemented on containers without a mandate and some clear, industry-wide, over-arching plan. Because of the number of organizations, both private and government, the data sharing, sharing cost, maintenance, deployment standardization, and numerous other issues, it is unlikely that such an integrated system would arise organically in the near future. This is a common challenge with using RFID as infrastructure: the benefits are spread across numerous organizations, but the costs are often concentrated to a few organizations. Equitably sharing costs and benefits can be an insurmountable challenge without a single dominant advocate. It is our opinion that technical solutions exist or can be easily developed in the near future to meet the technical requirements. The most significant problem is finding ways to incentivize organizations to work together to exploit the efficiencies in the system that are available.

5. Conclusions

In this report, we examine the use of passive RFID and location positioning within an intermodal facility. We find that the most promising technology is a combination of a system like the Mojix system and the KU Cargo tag or similar high gain antenna tag. We observed an early Mojix system demonstration that showed a combination of great potential and the challenges that are common with new systems: awkward configuration process, organizations unwilling to share information early on, and poor early-stage support. For example, the KU Cargo tag shows excellent capabilities, but is not yet commercially available primarily due to weak demand. We believe that the obstacles are part of the “growing pains” of an early-stage technology, but our opinion, there are no technical hurdles that are not insurmountable. We foresee that the largest hurdle will be the cross-organizational will to deploy such a system.

References

- [1] Daniel M. Dobkins. The RF in RFID. Newness, 2008.
- [2] “Wal-Mart Expands RFID Mandate,” RFID Journal, August 18, 2003, available <http://www.rfidjournal.com/article/view/539/1/1>.
- [3] International Organization for Standards, “Information technology – radio frequency identification for item management – part 6: Parameters for air interface communications at 860 MHz to 960 MHz,” ISO/IEC, Tech. Rep. 18000-6:2004/Amd 1:2006, 2006.
- [4] Supreetha Aroor and Daniel D. Deavours, Dual-Resonant Microstrip-Based UHF RFID Cargo" Tag. In Proc. IEEE MTT-S International Microwave Symposium 2008 (IMS2008) June 15-20, 2008, Atlanta, TX.
- [5] “Alien Higgs 3 Product Overview,” Alien Technologies, June, 2008, available http://www.alientechnology.com/docs/products/DS_H3.pdf.
- [6] Starport Technologies, “Products – Portunus | Starport Technologies,” <http://starporttech.com/products/portunus.html>, accessed November 20, 2008.
- [7] M. Eunni, M. Sivakumar, D. D. Deavours. A Novel Planar Microstrip Antenna Design for UHF RFID. *JSCI*, vol. 5, no. 1, January 2007, pp. 6-10.
- [8] Mojix. <http://www.mojix.com/>
- [9] Mojix, “Mojix eNode Family,” product brochure, 2008, available http://www.mojix.com/products/documents/Mojix_eNode_Brochure.pdf.
- [10] Mojix, “Mojix Star System,” product brochure, 2008, available http://www.mojix.com/products/documents/Mojix_STAR_System.pdf.
- [11] RF Controls home page, available <http://www.rfctrls.com/>
- [12] Uğur Sarac, F. Kerem Harmanci, and Tayfun Akgül, "Experimental Analysis of Detection and Localization of Multiple Emitters in Multipath Environments," *IEEE Antennas and Propagation Magazine*, Vol. 50, No. 5, October 2008.
- [13] Andy Holman, personal communications, November 2008.



Technical Report

Summary of Status:
A Unified Architecture for SensorNet with Multiple
Owners: Supplement to Advance SensorNet
Technologies to Monitor Trusted Corridors

University of Kansas, ITTC
V.S. Frost, G.J. Minden, J.B. Evans, L. Searl,
D.T. Fokum, D. Deavours, E. Komp, A. Oguna,
M. Zeets, M. Kuehnhausen, D. Depardo

EDS, an HP Company
J. Walther, L. Sackman, M. Gatewood,
J. Spector, S. Hill, J. Strand

ITTC-FY2010-TR-41420-15

July 2009

Project Sponsor:
Oak Ridge National Laboratory
Award Number 4000043403

Abstract

This effort is aimed at monitoring cargo movements along a trusted corridor, e.g., rail facilities, in association with an integrated data-oriented methodology to increase efficiency and security. This goal is being achieved by performing research and deployment of an associated testbed focused on rail transportation issues. This effort is being performed in conjunction with the private sector to enhancing the ability of the to efficiently embed security that provides business value such as safety, faster transport, and reduced theft while supporting law enforcement and national security. This has been demonstrated in a live “short haul” trial and will shortly be demonstrated in a “long haul” trial in a foreign country. (For background and definition of terms see [1]).

Table of Contents

Abstract	i
Table of Contents	ii
1. Executive Summary	1
2. Introduction	2
3. Status on Technology Proof of Concept and Integration of the SmartPort Trade Data Exchange and Transportation Security SensorNet Technologies	2
4. Status of the Development of Transportation Security SensorNet (TSSN) Technologies	2
5. Status of System Architecture, Modeling, and Optimization	3
6. Status of Communications System Evaluation	3
7. Status RFID Technology Evaluation and Development	4
8. Associated Efforts	4
9. Project Timeline	4
10. References	4

1. Executive Summary

This effort has had numerous thrusts, but is primarily focused on creating the Transportation Security SensorNet Technology (TSSN), information technology that bridges the gap between deployed sensor networks and decision-makers in a multi-modal, multi-organizational transportation environment. In January 2009, we successfully tested the system with a 20 km “short haul” rail-based test. This included the mating of sensor events and data from the Trade Data Exchange (TDE), complex event processing, and timely notification of events with relevant data to decision-makers. For this test the mobile rail network (MRN) was installed in a locomotive; sensors (electronic seals) were attached to several container cars and others placed in the locomotive; the train then traveled for five hours along an approximately 20 km route. KU/ITTC and HP (formerly EDS) personnel traveled with the MRN in the locomotive. The MRN communicated with the virtual operations center (VNOC) located at KU/ITTC in Lawrence, Kansas. The VNOC sent out alarm messages (both SMS and e-mail) to decision makers; in addition to sensor state and location, these messages contained logistics information obtained in real time from the TDE located in Overland Park, Kansas. Extensive message logs were recorded during this test; a visualization tool was developed to graphically study the logged messages and their timing. We have performed extensive analysis, resulting in several reports and publications [1-5], including several peer-reviewed under review and in preparation.

Since the successful short haul test, we have set three goals to be implemented for the next “long haul” test:

- Integrating satellite communications and asynchronous message passing into SOA
- Enhance sensor capability
- Implement security within the SOA

First, integration of satellite communications is straightforward, but the challenge is that most SOA tools assume a continuous (always-on) communications capability and will block waiting for a reply. Consistent with our implementation philosophy, we have found a solution that uses standards-based, open source-supported transport “protocol” that is compatible with SOA. Second, we investigated vendor product that provide enhanced “parent/child” capability and found that the products were too unreliable for use. Third, we have implemented security within portions of the TSSN, most notably between services that uses the publish/subscribe paradigm. We discovered that this is pioneering work, and have made contribution to the open source tools to support this task.

For the “long haul” test, we are partnering with a rail carrier to transport cargo from Mexico to the US. The trial will originate in Mexico, travel through Mexico to the US/Mexico boarder, cross the boarder, and terminate some place in the southeastern US (exact locations TBD). The exact date has not yet been decided, but will be in late July or early August. The objectives of the long haul test are to:

- Collect additional and more detailed system data during the long haul trial
- Demonstrate the new technical features described above
- Collaborate with stakeholders on technology demonstration to further demonstrate security, business value, and facilitate commercialization and impact

2. Introduction

This effort is aimed at monitoring cargo movements along a trusted corridor, e.g., rail facilities, in association with an integrated data-oriented methodology to increase efficiency and security. This goal is being achieved by performing research and deployment of an associated testbed focused on rail transportation issues. This effort is being performed in conjunction with the private sector to enhancing the ability of the to efficiently embed security that provides business value such as safety, faster transport, and reduced theft while supporting law enforcement and national security. This has been demonstrated in a live “short haul” trial and will shortly be demonstrated in a “long haul” trial in a foreign country. (For background and definition of terms see [1]).

3. Status on Technology Proof of Concept and Integration of the SmartPort Trade Data Exchange and Transportation Security SensorNet Technologies

The “short haul” trial that took place in January, 2009 demonstrated a full integration of the Trade Data Exchange (TDE) and the Transportation Security SensorNet Technology (TSSN). This was demonstrated when a sensor on the train indicated that the container door was opened: the TSSN was able to successfully interact with the TDE to extract manifest information which was combined with the sensor event, and the combined information was delivered to the decision-maker in a timely way (less than one minute). The “long haul” test scheduled for late July or early August will demonstrate further integration by implementing security between the TDE and TSSN.

4. Status of the Development of Transportation Security SensorNet (TSSN) Technologies

The development of the TSSN takes an SOA approach, building upon the original ideas of ACE but utilizing current technology and widely accepted open Web Service specifications and publicly available implementations which are suitable for Sensor Networks. Some of the Web Service specifications in use are SOAP, the WS-X specifications, OGC, and UDDIv3. The TSSN is being implemented in three phases. The first phase will be used in the field trials described above.

Phase 1 – Simple service messages based on OGC specifications (used in trials).

Phase 2 – Use full OGC specification interface messages.

Phase 3 – Use lessons learned from Phases 1 and 2 to make improvements.

Phase 4 – Satellite communications in conjunction with GSM and software support for asynchronous communications

Phase 5 – Security framework and implementation throughout the TSSN

Phases 1–3 have been completed, and lessons learned have been documented in [3] and [4]. Phase 4 has recently been completed and is documented in [4] and described more in Section 6. Phase 5 has partially been completed and has resulted in contributions to open source tools to support security. One of the major lessons learned is that the available open tool support is lacking the functionality for a fully generalized approach to

implementing security within the SOA. We have instead focused on implementing a fully-functional security model within the constraints of the contemporary tool support. Details will be documented in future reports.

5. Status of System Architecture, Modeling, and Optimization

This task is focused on developing models of the TSSN to identify trade-offs and enable system optimization. (An extended abstract of the model was presented in [5]. An updated description is under development.) The short-haul rail trial has provided the basis to determine the performance of the TSSN system with respect to detecting events on intermodal containers in a rail environment. Analysis of the data from the short-haul trial has provided realistic parameters, including message sizes, probabilities of notifying decision makers within a given interval, and network times from the train to a virtual network operations center, that will be used with the developed models to determine trade-offs and study system optimization. A full description of the short-haul trial and results of the post-processing is in [3].

The final step is to combine the developed models with the realistic parameters to conduct the trade-offs and system optimization. Specifically, the visibility of cargo shipments on a train will be determined as a function of sensors placement, onboard network and backhaul communication system; the system trade-offs and optimizations will be performed with respect to cargo visibility.

6. Status of Communications System Evaluation

Research is continuing on radio technologies for TSSN. As part of evaluating the current active container seal technology operating in the 916 MHz band, it was discovered that the communication range for the devices selected for this research was more limited than expected. We provided a custom bi-directional RF amplifier to their system, which boosted the communication range to over 400 meters, which was used for the “short haul” test.

We have investigated the use of a vendor-provided “parent-child” capability of the active seals. This capability would provide a second “parent” tag covertly located on the container, which would monitor the “child” seal. If the seal was broken or lost while the container was not in range of the mobile rail network, the parent would keep a status event, including the time of the event, which could later be recovered. After extensive testing, we determined that the vendor-provided capability was not sufficiently robust for inclusion on the “long haul” trial, and that the vendor-proposed fixes would not be available with sufficient lead time for the trial.

The mobile rail network communicates to the network operations center through either a GSM modem, or when there is no GSM coverage, a satellite modem. The satellite modem has been integrated into the mobile rail network. The substantial challenge has been the reliable exchange of messages over an unreliable link because the majority of SOA open tools assume a continuous (always-on) communications channel. The implementation uses using a Java Message Service (JMS) and the Apache ActiveMQ implementation at the protocol layer. This new capability will be exercised in the “long haul” test. The developed system is an example of a delay tolerant network (DTN) and

demonstrates that Java Message Service (JMS) and the Apache ActiveMQ implementation addresses DTN issues.

7. Status RFID Technology Evaluation and Development

The combination of the new (patent-pending) ITTC/KU on-metal RFID tag technology and the Mojix [6] system was deployed and tested in a warehouse environment. While this initial testing focused on the suitability of the system on an MES (manufacturing and execution system, i.e., an assembly line) and for scanning entering and exiting a dock door, the results of this testing lead to conclusions concerning applicability of the technology in an intermodal environment. Additional experiments have been conducted; the results of those experiments as well as the suitability for intermodal environments are described in [7].

8. Associated Efforts

KC SmartPort has continued to coordinate meetings for the groups involved in TSSN, CTIP and EFM, as well as additional activities. These meetings are creating a common, open environment with low entry barriers to enable broader access by stakeholders while contributing a venue to commercialization. Recently, KCS SmartPort has initiated a small activity to expand the capabilities of the TSSN, specifically to enhance the capability to track the transfer of containers across organizational boundaries.

9. Project Timeline

The “short haul” field trial was completed in January of 2009. A “long haul” field trial is currently being scheduled and is anticipated to take place in late July or early August of 2009. The long haul trial will start in Mexico, cross the US – Mexico border, and terminate somewhere in the southern USA. The efforts associated with the summer modeling, communications, and RFID are planned to be completed by the end of Fall 2009 and interim reports describing these activities are currently available. Activities associated with SmartPort, EFM, and CTIP will continue until June 2010. The current date of completion for the effort is June 15, 2010.

10. References

- [1] V.S. Frost, G.J. Minden, J.B. Evans, L. Searl and D.T. Fokum, T. Terrell, L. Sackman, M. Gatewood, J. Spector, S. Hill, and J. Strand, “Status Update : A Unified Architecture for SensorNet with Multiple Owners: Supplement to Advance Sensor Technologies to Monitor Trusted Corridors,” ITTC-FY2009-TR-41420-10 August 2008.
- [2] Daniel T. Fokum “Optimal Communications Systems and Network Design for Cargo Monitoring”, Proposal for Ph.D. dissertation research, Department of Electrical Engineering & Computer Science, University of Kansas, December 2008.

- [3] D. T. Fokum, V. S. Frost, D. DePardo, M. Kuehnhausen, A. N. Oguna, L. S. Searl, E. Komp, M. Zeets, D. D. Deavours, J. B. Evans, and G. J. Minden, "Experiences from a Transportation Security Sensor Network Field Trial," Information and Telecommunication Technology Ctr., University of Kansas, Lawrence, KS, USA, Technical Report ITTC-FY2009-TR-41420-11, June 2009.
- [4] M. Kuehnhausen, D. T. Fokum, V. S. Frost, D. DePardo, A. N. Oguna, L. S. Searl, E. Komp, M. Zeets, D. D. Deavours, J. B. Evans, and G. J. Minden, "Service Oriented Architecture for Monitoring Cargo in Motion Along Trusted Corridors," ITTC-FY2010-TR-41420-13 July 2009.
- [5] D.T. Fokum, "Optimal Communications Systems and Network Design for Cargo Monitoring," to appear Proc. Tenth Int'l Workshop on Mobile Computing Systems and Applications (HOTMOBILE 2009), Santa Cruz, California, USA
- [6] Mojix. <http://www.mojix.com/>
- [7] Daniel D. Deavours, "Application of Passive UHF RFID in Intermodal Facilities," ITTC-FY2010-TR-41420-14, July 2009.
- [8] V. S. Frost, G. J. Minden, J. B. Evans, "Summary of Status: A Unified Architecture for SensorNet with Multiple Owners: Supplement to Advance SensorNet Technologies to Monitor Trusted Corridors," ITTC-FY2010-TR-41420-12 December 2008.



Technical Report

EDS HP Final Report

EDS, an HP Company
J. Walther, L. Sackman, M. Gatewood,
J. Spector, S. Hill, J. Strand

ITTC-FY2010-TR-41420-16

December 2009

Project Sponsor:
Oak Ridge National Laboratory
Award Number 4000043403

0 Amendment History

Append document status to the following table (for example, Draft, Final or Release #).

CR# (optional)	Document Version#	Approval Date	Modified By	Section, Page(s) and Text Revised
	0.1	09 Feb 2009	Sackman, Larry	Initial Creation

CONTENTS

0	Amendment History.....	i
	Table of Contents	1
	List of Tables	2
1	Project Process	1
2	Project Deviations.....	2
	2.1 Schedule Modifications	2
	2.2 Budget Modifications	2
	2.3 Operational Field Tests	3
	2.3.1 Original Field Tests	3
	2.3.2 Project Field Tests	4
	2.3.3 Final Field Tests.....	5
3	Trade Data Exchange	6
	3.1 Overview	6
	3.2 Lessons Learned	6
	3.3 Future Directions	7
4	Acknowledgements	8
5	Legal Trademark Requirements.....	9

TABLES

Table 1	Original Planned and Finish Date for Each Phase.....	2
Table 2	Modified Project Schedule to Reflect the Planned Dates	2
Table 3	Original Budget Allocation, Changes Allowed, and Resulting Project Budget	3

1 Project Process

The purpose of the EDS effort was to produce an integrated SensorNet and Trade Data Exchange architecture and field tested in a rail environment of sensing prototype for intermodal transport. This goal was successfully achieved. An integrated SensorNet and Trade Data Exchange environment was designed, implemented, and demonstrated with two rail field trials. The effort resulted in two filed demonstrations of end-to-end operability of the use cases selected and defined with stakeholders (KCS).

For convenient reference, the following is a review of EDS' project process.

Milestone 1	Planning and Architecture Phase
	In this first phase, EDS formed the project team and engaged stakeholders to ensure their business and operational requirements were identified and documented. EDS finalized the project scope document, prepared the project management plan, finalized the business-technology processes and developed the enterprise architecture developed the test plan.
Milestone 2	Hardware Specification and Procurement Phase
	In the second phase, EDS specified and procured the servers to host the Trade Data Exchange (TDE) architecture and procured the Hi-G-Tek sensors. In addition, EDS developed the test scenarios for the operational field tests.
Milestone 3	SensorNet Configuration and Deployment Phase
	The third milestone involved SensorNet devices configuration, integration and deployment, plus core feature enhancement and system testing.
Milestone 4	TDE Integration and Testing Phase
	The fourth milestone involved integration of the SensorNet configuration with the SmartPort TDE architecture and two operational field tests: (1) a short-haul test in the KC metropolitan area and (2) a long-haul test.

2 Project Deviations

As the project progressed, the needs and requirements of stakeholders and the technologies selected necessitated modifications to EDS' original statement of work. A change order, Subcontract 2007-023 M1 executed in August 2008, redefined EDS' scope in terms of schedule, budget and operational tests. The following sections detail the modifications specified in the change order.

2.1 Schedule Modifications

EDS' original project schedule had a July 7, 2007, planned start date and a December 31, 2008, planned finish date. In accordance with the statement of work, the original schedule incorporated four deliverable-based milestones: (1) Planning & Architecture, (2) Hardware Specification & Procurement, (3) SensorNet Configure & Deploy and (4) TDE Integration/Testing. Table 1 shows the original planned and finish date for each phase.

Table 1

Deliverable	Original Planned Start	Original Planned Finish
Planning & Architecture	July 2007	September 2007
Hardware Specification & Procurement	October 2007	December 2007
SensorNet Configure & Deploy	January 2008	June 2008
TDE Integration/Testing	July 2008	December 2008

Following the August 2008 change order, EDS modified the project schedule to reflect the planned dates shown in Table 2.

Table 2

Deliverable	Project Planned Start	Project Planned Finish
Planning & Architecture	July 2007	September 2007
Hardware Specification & Procurement	October 2007	December 2007
SensorNet Configure & Deploy	January 2008	December 2008
TDE Integration/Testing	January 2009	August 2009

2.2 Budget Modifications

The original project budget allocated \$26,617.00 for computer equipment, \$253,378.00 for sensing devices and \$38,040.00 for travel. During Phase 2, Hardware Specification & Procurement, it became apparent that the original budget allocated more than would be required for computer equipment, sensing devices and for travel. Within the change order, KUCR and EDS agreed to reallocate budget dollars from these categories to labor. Table 3

shows EDS' original budget allocation, the changes allowed by the change order and the resulting project budget.

Table 3

Description	Original Budget	Change	Project Budget
Labor	\$430,875.00	\$155,721.00	\$586,596.00
Telecomm	\$1,500.00	\$3,500.00	\$5,000.00
Computer Equipment	\$26,617.00	(\$14,617.00)	\$12,000.00
Sensing Devices	\$253,378.00	(\$230,378.00)	\$23,000.00
Travel	\$38,040.00	(\$18,040.00)	\$20,000.00
TOTAL SUBCONTRACT	\$750,410.00	(\$103,814.00)	\$646,596.00

2.3 Operational Field Tests

A major aspect of the EDS contribution to this research was developing a relationship with the rail stakeholder, in this case KSC, defining the priority scenarios with the stakeholders, and coordinating and participating in rail field trials. While the specific nature of the field tests changed over time, two successful field trials were conducted: one in the US, and the second originating in Mexico and completing in the US. After reviewing customer requirements and priority business cases, EDS acquired the Hi-G-Tek sensors.

The original plan (2007) subcontract specified three field tests. However, the business requirements of stakeholders and technical limitations of sensors required that modifications be included in the change order. Section 0 details the original field test, and as the project evolved, a second more specific set of field test were defined. Section 0 details the modified project field tests. Section 3.3.3 contains an overview of the two field experiments that were conducted in January and July of 2009.

2.3.1 Original Field Tests

Test 1	International in-bound cargo through Mexico
	Equip three to five intermodal cargo containers will be equipped with tracking and sensing devices. Tracking and sensing information will potentially include data such as door open, door close, temperature, vibration, as well as chemical and radiation profiles. Stakeholders will include an international shipper, Kansas City Southern, Mexican and US brokers and a local truck line carrier, US Customs And Border Patrol, Mexican Customs, and the Port of Lazaro Cardenas.
Test 2	International out-bound cargo to Mexico

	Equip one to two tractor trailer rigs with tracking and sensing devices. Monitor the rigs they travel from Kansas City, through the Laredo, TX port to a final destination in Mexico. Tracking and sensing information will potentially include data such as door open, door close, temperature, vibration, as well as chemical and radiation profiles. Stakeholders will include an International shipper or third-party logistics (3PL), Mexican brokers, a truck line carrier and Mexican Customs.
Test 3	International in-bound cargo through US West Coast port
	Similar to Test 1, however, the route will be through a US West Coast port of entry (for example, Seattle/Tacoma or Los Angeles/Long Beach) and travel over the Burlington Northern Santa Fe (BNSF) rail line to Kansas City, MO, intermodal transfer to truck line and delivery to ultimate destination. Tracking and sensing information will potentially include data such as door open, door close, temperature, vibration, as well as chemical and radiation profiles. Stakeholders will include an International shipper or third party logistics (3PL), US Customs and Border Patrol, a US brokers and BNSF rail line.

2.3.2 Project Field Tests

Test 1	KC metropolitan short haul
	Equip one to three intermodal cargo containers with tracking and sensing devices. Tracking and sensing will potentially include sensing information such as door open and door close. Stakeholders will include Kansas City Southern Rail lines.
Test 2	Mexico port of entry to Nuevo Laredo, MX
	One to three intermodal cargo containers will be equipped with tracking and sensing devices and monitored as they travel from a Mexican port of entry to Nuevo Laredo. Tracking and sensing will potentially include sensing information such as door open and door close. Stakeholders will include Kansas City Southern de Mexico.
Test 3	Mexico port of entry to Guadalajara, MX
	One to three intermodal cargo containers will be equipped with tracking and sensing devices and monitored as they travel from a Mexican port of entry to Guadalajara. Tracking and sensing will potentially include sensing information such as door open and door close. Stakeholders will include Kansas City Southern de Mexico.

2.3.3 Final Field Tests

Regrettably, the economic recession of 2008 further restricted the participation of stakeholders. As a result, the number of test was reduced from three to two. The final tests executed under the subcontract are detailed below.

Test 1	KC metropolitan short haul
	Equip one to three intermodal cargo containers with tracking and sensing devices. Tracking and sensing include the following data: sensor present, sensor missing, GPS, and door open and door close. Stakeholders will include Kansas City Southern Rail lines. See the ITTC technical report ITTC-FY2009-TR-41420-11 for summary and analysis of the KC metropolitan short haul trial.
Test 2	San Luis Potosi, MX, to Nuevo Laredo, MX, to Laredo, TX, to US destination
	One to three intermodal cargo containers will be equipped with tracking and sensing devices and monitored as they travel from a Mexican port of entry to Nuevo Laredo. Tracking and sensing include the following data: sensor present, sensor missing, GPS, door open and door close. Stakeholders will include Kansas City Southern de Mexico. A full description and analysis of this trial will be available shortly in an ITTC technical report.

3 Trade Data Exchange

3.1 Overview

The Trade Data Exchange (TDE) contains commercial shipping data. The TDE is based on a standards-based, service-oriented architecture. Hosted on a server geographically separate from the VNOC, the TDE responds to queries from the VNOC. Finally, the TDE sends startMonitoring, stopMonitoring, and getLocation messages to the VNOC.

The TDE monitors the progress of shipments from the point of origination to the point of destination. The TDE captures commercial and clearance data including the shipping list, bill of lading, commercial invoice, certificate of origin and shipper's export declaration. The TDE validates data within and across these trade documents to ensure the data is accurate, consistent and complete. Further, the TDE will monitor the progress of the documentation and notify responsible parties when errors or incompleteness pose the threat of delaying a shipment. Finally, the TDE will also forward notification to the customs broker to request verification of the trade origination documents. The customs broker accesses the TDE via the same portal to review and verify the trade documentation. The TDE will also allow for collaboration between participating shippers, third-party logistics providers, carriers and customs brokers to define and document business requirements and risk assessment requirements.

The TDE was successfully integrated with SensorNet as demonstrated with the field trials. The TDE continues to be developed and integrated into the Kansas City SmartPort with expectations of becoming operational in 2010. The TDE and SensorNet projects continue to be further integrated with separate funding to include automated data collection of changes in custody of shipments.

3.2 Lessons Learned

All of the major goals of the project were met. We were able to successfully demonstrate the integration of the TDE and SensorNet systems on a realistic scenario that demonstrated business value to stakeholders.

Through the process, we encountered a number of situations that we have learned from. These are given in the table below.

Automate startMonitoring Activity	The manual startMonitoring activity required that additional resources be available at the beginning of the short-haul and long-haul test. As a result, resources were providing support at atypical working hours. Further, the long-haul test was a 24x7 operation throughout its duration. An automated startMonitoring action would have benefited those on the train and KU ITTC resources in Lawrence during the operational tests.
--	---

Secure Greater Commitment from Operational Test Stakeholder	KCS played a vital and critical role in the ability to demonstrate the integration of the TDE with the SensorNet through extensive facilitation efforts associated with both operational tests. To their credit, KCS did provide valuable resources (using only their internal resources) in people, access and equipment. For this, the project team is grateful. However, a stronger relationship with the stakeholder may have fostered a greater investment on the part of KCS, which is critical for the long-term impact of the developed systems.
MCS Data	Moving forward, an automated feed of MCS data, rather than static data formulated by the project team, would have allowed the operational tests to reflect a better, real-world type of scenario.
Sensor Hardware	<p>The sensor hardware from Hi-G-Tek functioned as expected during both operational tests with one exception, the parent/child capability. Hi-G-Tek was selected as the sensor hardware provider because their technology was marketed and sold as having the parent/child capability. The project team did not learn that Hi-G-Tek's hardware available in 2008 and 2009 did not include the parent/child capability.</p> <p>EDS worked with Hi-G-Tek to incorporate the parent/child capability into the sensor hardware procured for the project. Hi-G-Tek, however, was not able to complete implementation of the capability without adversely affecting the schedule of the long-haul test.</p>

3.3 Future Directions

Continued Partnership	<p>HP sees great value in continuing the partnership between KU ITTC and EDS. We see the SensorNet technology becoming an integrated component of the TDE solution. Using SensorNet as the gateway between field-deployed sensing technologies and the TDE allows the TDE to offer services wider audience of transportation industry service providers.</p> <p>That SensorNet keeps the TDE agnostic of sensors deployed in the field is a value proposition to all. Further, that the SensorNet can provide sensor data related to field events allows transportation stakeholders the ability to more closely monitor cargo and assets. This in turn can help the industry minimize loss due to theft and tampering, which provides a means to stakeholders to operate more efficiently in such a competitive industry.</p>
Continued Development of the TDE	Related to the TDE specifically, EDS would very much like to keep the solution moving forward to its eventual objective: a commercial product. To that end, EDS continues to pursue sources of funds to continue development of the product.

4 Acknowledgements

The authors would like to acknowledge Kansas City Southern de Mexico for their vital participation in the long-haul rail trial; specifically, Jim Kneistadt, Head of Security, and Alan Martinez.

5 Legal Trademark Requirements

EDS

EDS and the EDS logo are registered trademarks of Hewlett-Packard Development Company, LP. HP is an equal opportunity employer and values the diversity of its people.
© 2009 Hewlett-Packard Development Company, LP.



Technical Report

**TRANSPORTATION SECURITY SENSOR NETWORK:
SENSOR SELECTION AND SIGNAL STRENGTH ANALYSIS**

Angela Oguna

ITTC-FY2010-TR-41420-17

December 2009

Project Sponsor:
Oak Ridge National Laboratory

Abstract

Cargo theft is a major problem in the US; the FBI estimated losses of \$15-30 billion in 2006. The Transportation Security Sensor Network (TSSN) aims to mitigate these risks by utilizing sensors that will track and monitor train-borne shipping containers. Prior to deployment, we need to know the read ranges of proposed sensors and their practicability in a rail scenario. I describe the experiments that were performed to test the sensors; results indicate that the wire sensor is the most suitable. Ultimately, I expect that the improved cargo security resulting from TSSN implementation will lead to fewer incidences of theft, thereby lowering prices for the final consumer.

CONTENTS

I	Introduction	1
II	System Architecture	1
II-A	Mobile Rail Network	1
II-B	Virtual Network Operations Center	3
II-C	Trade Data Exchange	3
III	Tests	3
III-A	Read Range Tests	3
III-A1	Free Space Tests	3
III-A2	Tests in the presence of a Ground Plane	4
III-A3	Test with Cars	4
III-A4	Test on a Trailer	4
III-B	Short-haul Rail Trial	4
IV	Results	4
IV-A	Read Range Tests	5
IV-A1	Free Space Tests	5
IV-A2	Tests in the presence of a Ground Plane	5
IV-A3	Test with Cars	5
IV-A4	Test on a Trailer	6
IV-B	Short Haul Train Test	6
IV-B1	TSSN Component Interaction	6
IV-C	HSDPA Signal Strength	6
V	Conclusion	7
	Acknowledgments	7
	References	8

LIST OF FIGURES

1	Transportation Security Sensor Network (TSSN) Architecture	2
2	Mobile Rail Network Hardware	2
3	Wireless Sensors	3
4	Read Range Success versus Seal-Reader separation in free space	5
5	Read Range Success vs. Seal-Reader Separation	5
6	Read Range Success versus Seal-Reader Separation in Presence of Cars	6
7	HSDPA Signal Strength vs. Time	7

LIST OF TABLES

I	Number of Messages Generated by TSSN Components	7
---	---	---

I. INTRODUCTION

CARGO shipments are subject to theft, hijacking, and tampering. In 2006 the FBI estimated that cargo theft cost the US economy between \$ 15–30 billion in annual losses [1]. However, law enforcement acknowledges that these values are only about 40% of the losses that occur; due to the reluctance of businesses to report theft. Cargo is also used as a guise to transport illegal drugs, arms, and aliens; giving rise to other forms of crime that law enforcement officers tackle daily. Indirect costs stemming from cargo crimes, such as delayed deliveries, insurance claims and processing, and in the worst case scenarios injuries or loss of life, result in total losses that are 4-5 times greater than the direct losses [2]; a cost that the consumer eventually carries.

Cargo transportation requires a complex interaction between the originator, the shipper and the receiver. This paper describes two components of a system designed to minimize the effects of cargo crime. The transmission ranges of the sensors used will be measured to design and deploy the system. The Global System for Mobile (GSM) communications signal strength along a train route will be collected to guide the future design of algorithms that switch between communication routes.

A hardware and software system referred to as the Mobile Rail Network (MRN) monitors the cargo in transit. The Mobile Rail Network sends alerts to the Virtual Network Operations Center (VNOC), which processes the messages to determine if the shipper and/or recipient should be notified. The VNOC communicates with the Trade Data Exchange (TDE) to get information on the cargo shipment, and determine the personnel to be informed of the security alert. Therefore, the Mobile Rail Network, Virtual Network Operations Center, and the Trade Data Exchange link the originator, the shipper and the receiver; ensuring that informed decisions can be made in a timely manner in case of a security breach.

This paper describes the component interaction within the TSSN and experimental data documenting suitable hardware for a rail environment. The results show that the TSSN can effectively monitor cargo, and notify decision makers of security breaches. The rest of the paper is laid out as follows: Section II , describes the TSSN architecture and its components. Section III discusses two experiments to determine suitable hardware for a rail environment, and also assesses the effectiveness of the TSSN system in cargo monitoring. Section IV describes the results of our tests and finally Section V describes the conclusion.

II. SYSTEM ARCHITECTURE

A Transportation Security Sensor Network (TSSN) was set up to achieve the objectives stated above. The TSSN utilizes a Service Oriented Architecture (SOA) to provide a reusable framework that can be implemented across the transportation industry [3]. It uses open web standard interfaces, such as Apache Axis 2, to process and share information across different applications. The main components of the TSSN are the Mobile Rail Network (MRN), Virtual Network Operations Center (VNOC), and the Trade Data Exchange (TDE), which allow interaction between the originator, shipper, and receiver as illustrated in Fig. 1.

A. Mobile Rail Network

The Mobile Rail Network (MRN) includes the software and hardware that monitor freight on the train and report any suspicious activity to a Virtual Network Operations Center (VNOC). The hardware component of the MRN consists of a set of wireless shipping container security sensors positioned on individual containers, an electronics suite located in the locomotive, and a set of antennas that is magnetically mounted on the locomotive roof to maximize reception. The electronics suite contains a computing platform, a power inverter, a three-axis accelerometer, a security seal interrogation transceiver and wireless data modems as illustrated in Fig. 2

The MRN software consists of the MRN SensorNode, the MRN AlarmProcessor and a communications service. If the seals are tampered with, they send an alert burst message to the MRN SensorNode. The MRN SensorNode service determines the seal events that are unsafe and it sends an alert message to the MRN AlarmProcessor service for each suspicious event. The MRN AlarmProcessor performs further

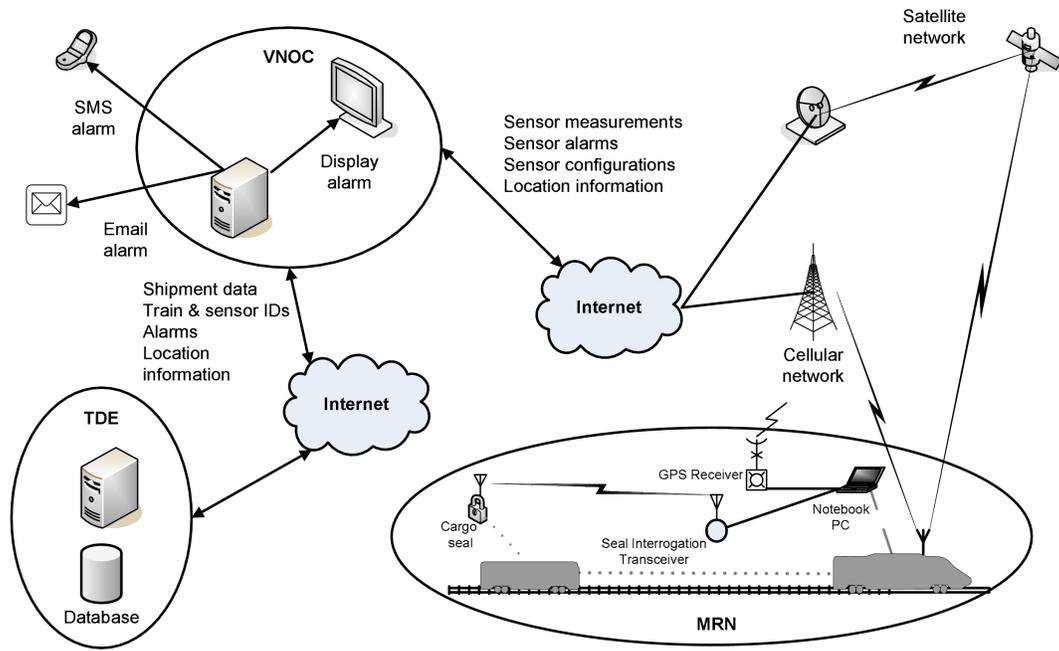


Fig. 1. Transportation Security Sensor Network (TSSN) Architecture

processing on the alert and sends an MRN alarm message to the VNO AlarmProcessor if the event is indeed unsafe. The communications service logs the High Speed Downlink Packet Access (HSDPA) signal strength-information that will determine when the communications system should switch between the Iridium satellite and Global System for Mobile (GSM) communication connection.

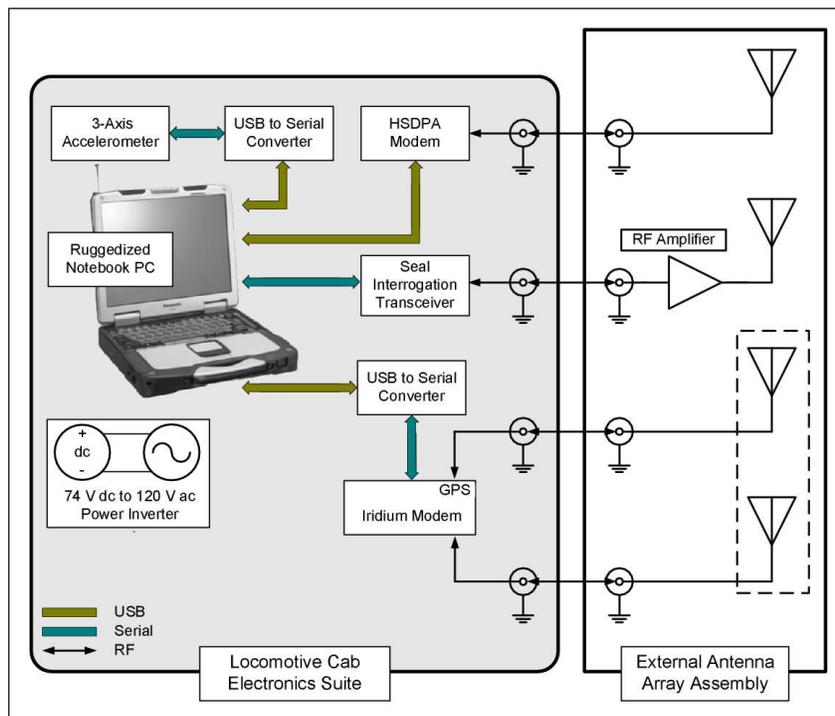


Fig. 2. Mobile Rail Network Hardware

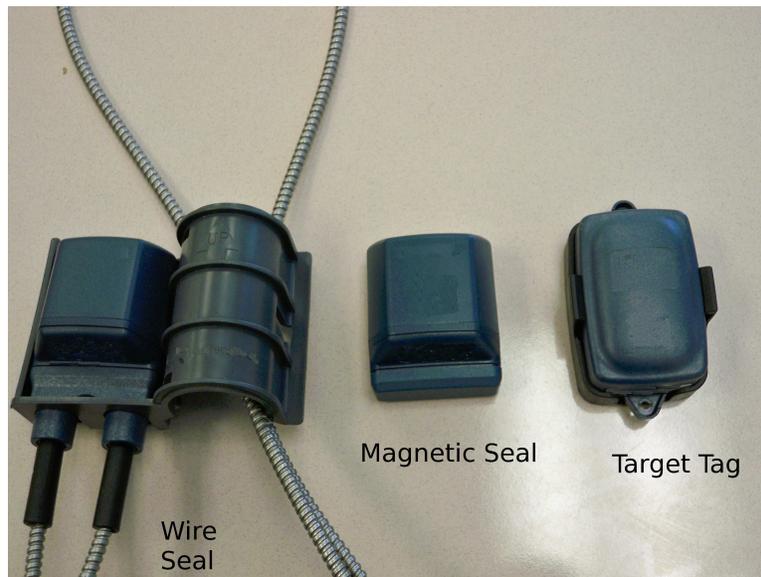


Fig. 3. Wireless Sensors

B. Virtual Network Operations Center

The Virtual Network Operations Center (VNOC) contains a VNOC AlarmProcessor and a VNOC AlarmReporting Service which run on a remote server at the monitoring location. The VNOC AlarmProcessor receives alarms from the MRN AlarmReporting Service. It queries the Trade Data Exchange for cargo information and uses the response to determine if a VNOC alarm should be forwarded to personnel. The MRN alarm and shipment data are combined into a message that is sent to the user by the VNOC AlarmReporting Service via email and/or SMS. The VNOC also transmits startMonitoring, stopMonitoring, and getLocation commands from the TDE to the MRN.

C. Trade Data Exchange

The Trade Data Exchange (TDE) contains the cargo information, which it relays when queried by the VNOC about specific shipments. It also stores alarm messages sent to the user by the VNOC in addition to sending startMonitoring, stopMonitoring, and getLocation commands to the MRN via the VNOC.

III. TESTS

Two sets of experiments were conducted to test the TSSN hardware. The first set of experiments analyzes the performance of the sensors in a static environment, while the second set of tests records the High-Speed Downlink Packet Access (HSDPA) signal strength and evaluates the overall TSSN performance.

A. Read Range Tests

The primary objective of this test was to determine which wireless seal provided the best read ranges, and would best withstand a rail environment. The magnetic seal, target tag and the wire seal, illustrated in Fig. 3, were tested in free space, in the presence of metal ground plane, near cars and next to a trailer.

1) *Free Space Tests*: The seal interrogator transceiver (SIT) and the magnetic seal were placed on plastic carts to elevate them and minimize grounding effects. The SIT antenna was placed at a fixed position, while the seal was moved away from the SIT antenna in 10 m increments. The seal was interrogated ten times at each new seal position, and then the seal-SIT antenna separation was incremented by a distance of 10 m. Responses received within two minutes were recorded as successes; otherwise they were counted

as failures. The procedure was repeated until the maximum read distances for each seal was reached or exceeded.

A line of sight path existed between the SIT antenna and the seal during the test. Both the SIT reader and the laptop remained powered for all the tests, except for the test performed on the trailer because their battery power could only last 1.5 hours before shutting off.

2) *Tests in the presence of a Ground Plane:* The objective of this test was to determine the effect of a metal ground plane on the read range. The same procedure outlined in free range test was followed, but the seal interrogator transceiver was placed on 1.5 m \times 0.9 m metal sheet. Ten readings were taken as before and the tests were repeated until the maximum read range was obtained, or the number of successful readings fell below two. The tests were repeated with the SIT antenna placed on the metal sheet, and the seal positioned on a Styrofoam block covered with aluminum foil to determine the effects of placing both the seal interrogator transceiver and the seal on metal ground planes.

3) *Test with Cars:* The objective of this test was to test the effect of large interfering metal objects on the read ranges. The seal interrogator transceiver was positioned on a cars trunk lid. The seal was placed on a 0.9 m high wooden block one car width away from the reader. Unlike previous tests, there was no direct line of sight path between the seal interrogator and the seal. Ten readings were taken as before at each position, and the seal was then moved one parking spot (2.5 m) farther away. The tests were repeated until the number of successful readings dropped to two.

4) *Test on a Trailer:* The final test was performed on a 16 m trailer to simulate a rail environment. A car was parked in front of the trailer and the SIT antenna was placed on the cars roof while the seal was placed at the back of the trailer. Ten queries were sent out by the seal interrogator as in the previous tests and if no response was received within two minutes, the interrogation was counted as a failure. There was no line of sight path between the seal and the seal interrogator, and both the seal interrogator and laptop were running on battery power.

B. Short-haul Rail Trial

This test was carried out on a train traveling on a 35 km route from an intermodal shipping facility to a rail yard. The main objectives were to analyze message transmission between the TSSN components for correctness and monitor the HSDPA signal strength to determine the feasibility of switching between an Iridium satellite and HSDPA link to relay messages between the MRN and the VNOC. During the test, the VNOC was located at the university (approximately 60 km away), the TDE was at a remote location approximately 48 km away, and the MRN was located in the locomotive cab. Several seals were hung on intermodal containers, and one seal was kept in the locomotive cab with the MRN electronics suite. The latter seal was opened and closed to simulate seal open and close events. The VNOC AlarmProcessor received alerts from the MRN which contained the event time, seal position, message type, unique sensor ID and the event type. The VNOC AlarmProcessor queried the TDE to obtain the shipment information and decided (based on a set of rules) if personnel should be notified. If the alarm met the set criteria, the MRN alert and the shipment data were combined into an email or SMS message that was sent to the user by the VNOC AlarmReporting service. The GSM signal strength between the MRN and the VNOC was monitored and recorded in log files by the communications service. The experiment was considered a success as all the events detected by the seals were processed and reported to the personnel using email and SMS.

IV. RESULTS

This section discusses the results of the TSSN hardware evaluation and the HSDPA signal strength experiment, in addition to presenting brief results on overall TSSN performance.

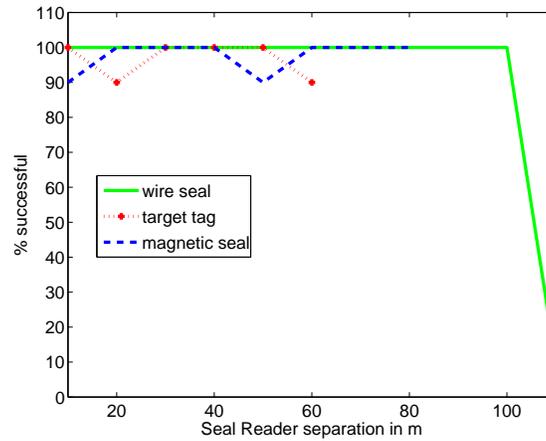


Fig. 4. Read Range Success versus Seal-Reader separation in free space

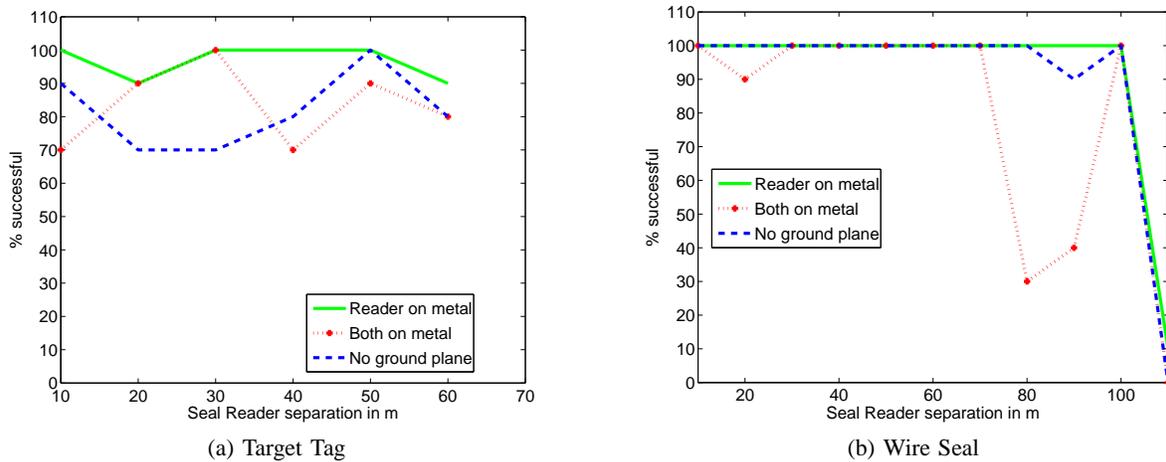


Fig. 5. Read Range Success vs. Seal-Reader Separation

A. Read Range Tests

1) *Free Space Tests:* The highest percentage of successful readings was obtained in the free space tests which were characterized by a direct line of sight in the duration of testing. The wire seal had the longest read range (100 m) followed by the magnetic seal which could be read up to 90 m, contrary to manufacturers specified range of 50 m. The target tag achieved its stated read range of 50 m. The results illustrated in Fig. 4 show the superior performance of the wire seal, as it recorded 100% success rate throughout the test.

2) *Tests in the presence of a Ground Plane:* The performance of both the magnetic seal and the target tag seals deteriorated when a ground plane was introduced in the testing environment. Their performance further declined when both the SIT antenna and the seals were placed on ground planes in comparison to the scenario where only the SIT was positioned on a ground plane. Fig. 5a illustrates the performance of the target seal when tested with and without a ground plane.

The wire seal performance was not affected greatly by the ground plane especially at shorter distances as shown in Fig. 5b. Although a lower performance is noted as the seal approaches its maximum read range, it clearly displays a superior performance when compared to the target tag and magnetic seal.

3) *Test with Cars:* The wire seal and target tag performed well at short distances when tested with cars. However, their performance declined as the separation distance was increased. A lower success rate had been expected for the target tag and magnetic seal due to the ground planes introduced by the car bodies. The poor performance for the wire seal could be attributed to the lack of a line of sight path in

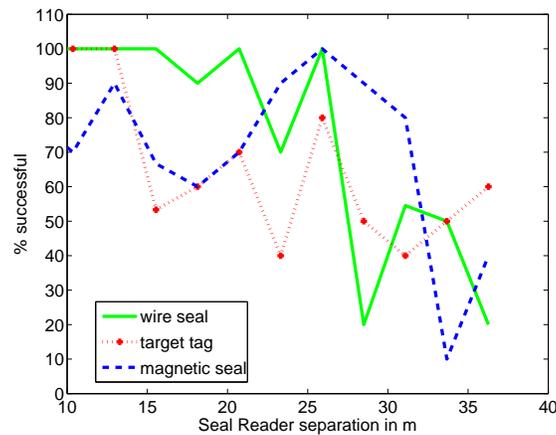


Fig. 6. Read Range Success versus Seal-Reader Separation in Presence of Cars

this experiment. Furthermore, the tests were carried out in an open parking lot, while the previous tests had been performed next to several large buildings. This indicates that the seal performances in the initial tests may have been improved by signal reflection from the surrounding buildings. Fig. 6 displays the performance of the seals in the presence of cars.

4) *Test on a Trailer:* The final test involved testing the seals when attached to the end of a 16 m (53') trailer. This time there was no line of sight path between the seal and the seal interrogator, and the open location minimized reflection from surrounding buildings. The percentage of successful readings recorded with the target tag, wire seal and magnetic seal were 0%, 20%, and 10% respectively. The decreased performance was partly attributed to the absence of a line of sight path between the SIT reader and the seal, as well as a lower transmit power since the seal interrogator transceiver was running on battery power. However, more tests are needed to confirm our assertions. The wire seal displayed a better performance than the magnetic seal and the target tag; making it most suitable for the rail environment.

B. Short Haul Train Test

1) *TSSN Component Interaction:* Table I shows the messages that were transmitted between several TSSN components. All VNOc queries were responded to by the TDE, i.e., 63 shipmentQueries and 63 shipmentQueryResponses, and the MRN responded to all commands from the TDE which were sent via the VNOc. This illustrates that all three TSSN components could communicate successfully without messages being dropped. Table I also illustrates that some messages are filtered by the system. The MRN SensorNode reported 546 alerts to the MRN AlarmProcessor. Only 131 alerts met the criteria set by the rules in the MRN AlarmProcessor and were sent out as MRN alarms to the VNOc AlarmProcessor. All the MRN alarms were sent out as VNOc alarms; indicating that they met the set criteria, and were therefore, sent out as SMS or email messages to decision makers [4]. Our results confirmed that the TSSN could not only detect unsafe events, but it could process the messages and relay the information to decision makers.

C. HSDPA Signal Strength

In the current TSSN implementation the MRN is instructed, at startup, to either use an Iridium satellite or HSDPA link to transmit messages between the VNOc and the MRN. Future TSSN implementations will have an algorithm that can switch dynamically between HSDPA and satellite link transmissions.

The MRN communications service monitored HSDPA signal strength during the short-haul trial. Fig. 7 provides a trace of the change of signal strength with time. The signal strength is constant at the beginning of the trip. This corresponds to the time when the MRN was on, but the train was stationary. There are

TABLE I
NUMBER OF MESSAGES GENERATED BY TSSN COMPONENTS

Message Type	From	To	No. of Messages
Alerts	MRN SensorNode	MRN AlarmProcessor	546
MRN Alarm	MRN AlarmProcessor	VNOC AlarmProcessor	131
VNOC Alarm	VNOC AlarmProcessor	VNOC AlarmReporting	131
getLocation	TDE	MRN SensorNode	30
Location	MRN SensorNode	TDE	30
shipmentQuery	VNOC AlarmProcessor	TDE	63
shipmentQueryResponse	TDE	VNOC AlarmProcessor	63

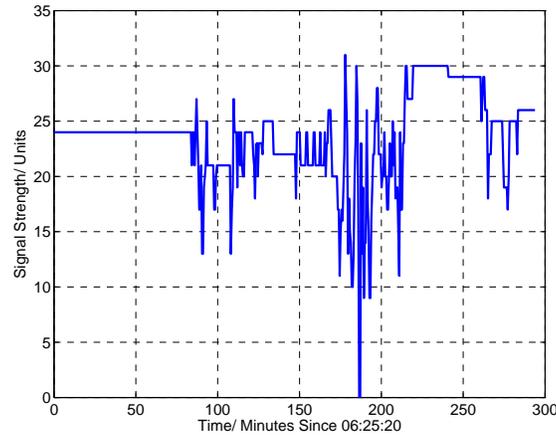


Fig. 7. HSDPA Signal Strength vs. Time

two other constant portions at about 220 and 240 minutes when the train stopped at a train crossing. Once the train journey begins, the signal strength varies between 18 and 24 units. For the greater part of the journey, the signal strength is reliable except at the 180th minute when the signal strengths drops to 0. This illustrates the viability of a dual communication system that switches between the HSDPA and Iridium link in areas with a strong HSDPA signal strength. The more expensive iridium satellite could be turned on in areas with low HSDPA signal.

V. CONCLUSION

This paper presents results from hardware testing and a short haul trial of the Transportation Security Sensor Network (TSSN). The wire seal was the most practical sensor for a rail environment since it had a long read range, and was not affected greatly by metal surfaces. A strong HSDPA signal was detected along the train route, although this result does not generalize to other train routes, it was useful information for interpreting other results of this experiment. In addition, the collected HSDPA signal strength data will be used in the future design of an algorithm that can dynamically switch between modes of communication. Although the TSSN system can effectively monitor cargo, and transmit messages to decision makers; a lot of messages were dropped at the MRN AlarmProcessor. Given the large number of filtered messages, it is important to perform further analysis to see if the rules that determine unsafe events are incorrectly dropping important messages. This early test of the TSSN provides evidence that this design can be efficient in streamlining the communication between the originator, shipper, and the recipient to ensure safer cargo transportation. By reducing the risks of cargo theft, we hope that this will result in monetary savings for manufacturers that will eventually trickle down to the final consumer through lower prices.

ACKNOWLEDGMENTS

This work was supported in part by Oak Ridge National Laboratory (ORNL)—Award Number 4000043403.

REFERENCES

- [1] Federal Bureau of Investigation. (2006, July 21) Cargo Theft's High Cost. Headline. Federal Bureau of Investigation. [Online]. Available: http://www.fbi.gov/page2/july06/cargo_theft072106.htm
- [2] R. J. Fischer and G. Green, *Introduction to Security*, 7th ed. Boston, MA: Butterworth-Heinemann, 2004.
- [3] K. Martin, "Service Oriented Architecture for Monitoring Cargo in Motion along Trusted Corridors," University of Kansas, Lawrence, KS, ITTC Tech. Rep. ITTC-FY2010-TR-41420-13, July 2009.
- [4] D. T. Fokum *et al.*, "Experiences from a Transportation Security Sensor Network Field Trial," University of Kansas, Lawrence, KS, ITTC Tech. Rep. ITTC-FY2009-TR-41420-11, June 2009.



Technical Report

Application of the Java Message Service in Mobile Monitoring Environments

Martin Kuehnhausen and Victor S. Frost

ITTC-FY2010-TR-41420-18

December 2009

Project Sponsor:
Oak Ridge National Laboratory
Award Number 4000043403

Table of Contents

Table of Contents	i
List of Figures	i
List of Tables	ii
Abstract	1
I. Introduction	1
II. Problem Area	1
A. Asynchronous Communication	2
B. Message Security and Integrity	2
C. Scalability	2
III. Related Work	2
A. Java Message Service	2
B. Web Services	2
IV. Proposed Solution	3
A. Java Message Service	3
B. Transportation Security SensorNet	4
V. Results	8
A. Stationary	9
B. Mobile	9
VI. Conclusion	9
Acknowledgment	10
References	10

List of Figures

Figure 1: JMS administration according to [22] single consumer that have not been received. ...	4
Figure 2: Point-to-Point messaging	4
Figure 3: Publish/Subscribe messaging when the message is sent and vice versa.	4
Figure 4: TSSN physical architecture adapted from [25]	5
Figure 5: Mobile Rail Network message overview from [20] specifications.	5
Figure 6: Virtual Network Operation Center message overview from [20]	6
Figure 7: JMS transport receiver configuration in axis2.xml	7
Figure 8: JMS transport sender configuration in axis2.xml	7
Figure 9: JMS service queue name configuration in services.xml	7
Figure 10: ActiveMQ broker configuration in activemq.xml	7
Figure 11: ActiveMQ transport connector configuration in activemq.xml	7
Figure 12: ActiveMQ MRN network connector configuration in activemq.xml	8
Figure 13: One-way JMS message transmission messages to be forwarded in both directions... ..	8
Figure 14: Two-way JMS message transmission	8
Figure 15: Route for longhaul field trial, route starts at San Luis Potosi and ends approximately 210 miles down the track	8
Figure 16(a): ActiveMQ message queue for stationary scenario Initial test	9
Figure 16(b): ActiveMQ message queue for stationary scenario Follow-up tests	9
Figure 17: ActiveMQ message queue for mobile scenario	10

List of Tables

Table I: Elapsed Time From MRN to VNOC During Trial in Seconds	9
Table II: Comparison of Elapsed Time From MRN to VNOC of Longhaul to Shorthaul Trial in Seconds	9

Application of the Java Message Service in Mobile Monitoring Environments

Martin Kuehnhausen, *Graduate Student Member, IEEE* and Victor S. Frost, *Fellow, IEEE*

Abstract—Distributed systems and in particular sensor networks are in need of efficient asynchronous communication, message security and integrity, and scalability. These points are especially important in mobile environments where mobile remote sensors are connected to a control center only via intermittent communication, e.g. via satellite link. We present an approach that is able to deal with the issues that arise in such scenarios. In particular we focus on providing a solution that allows for flexible and efficient cargo monitoring on trains.

The Java Message Service presents a flexible transport layer for asynchronous communication that provides a transparent *store-and-forward* queue mechanism for entities that need to be connected to each other. Previously JMS was primarily used in always-connected high-bandwidth enterprise communication systems. We present the advantages of using JMS in a mobile bandwidth limited and intermittently connected monitoring environment and provide a working implementation called the Transportation Security SensorNet (TSSN). It makes use of an implementation of JMS called ActiveMQ that is used here to enable monitoring of cargo in motion along trusted corridors.

Results obtained from experiments and a field trial show that using JMS provides not just a practical alternative to often custom binary communication layers but a better and more flexible approach. One reason for this is transparency. Applications on both communication ends only need to implement JMS connectors while the remaining functionality is provided by the JMS implementation. Another benefit arises from the exchangeability of JMS implementations.

In utilizing JMS we present a new and flexible approach to deal with challenges such as intermittent and low-bandwidth communication in mobile monitoring environments.

Index Terms—Telemetry, Transport protocols, Intermittently connected wireless networks, Communication system software, Data communication, Software engineering

I. INTRODUCTION

THE primary use of Java Message Service (JMS) is in always-connected high-bandwidth enterprise communication systems but its concepts and techniques are useful in other scenarios as well. This paper describes the application of JMS in mobile monitoring environments.

One of the main advantages of using JMS is the fact that applications do not need to be modified to implement their own *store-and-forward* or *resend* mechanisms. How this is achieved is explained in detail later. Furthermore an example of an implementation of a mobile monitoring system is given that was field tested in stationary as well as intermittent mobile scenarios.

M. Kuehnhausen and V. S. Frost are with the Information and Telecommunication Technology Center, The University of Kansas, Lawrence, KS, 66045, USA; Corresponding author: mkuehnha@itc.ku.edu

This work was supported in part by Oak Ridge National Laboratory (ORNL)—Award Number 4000043403. This material is also partially based upon work supported while V. S. Frost was serving at the National Science Foundation.

II. PROBLEM AREA

Whenever disparate systems are deployed in the field that need to communicate with each other and a control center, there exist particular problems that need to be addressed. Here we used the following scenario as a motivating example.

Sensors are connected to cargo containers which they monitor. A train is then used to transport these containers. The sensors have limited capabilities and are managed locally by a more powerful sensor node which has extended functionality including a communication link back to a control center. Whenever a sensor detects an event it notifies the sensor node immediately. The sensor node then performs a simple evaluation of the event and decides whether or not to send it to the control center. In this paper we focus on the part that comes next, sending messages to and receiving control messages from the control center.

The communication link used may provide only intermittent communication. Therefore, the sensor node must deal with establishing the connection as well as transmitting messages. Especially the latter can cause problems. In a synchronous communication model the sensor node would only be able to send one message at a time and block while waiting for its acknowledgement. This is not feasible in this case because of the intermittent connection, low bandwidth and high latency of the communication link. An asynchronous communication model overcomes this blocking problem and is therefore more suitable. Furthermore, since messages cannot be sent out immediately due to the intermittent connectivity they need to be stored. This is often done by implementing a queuing mechanism inside of the sensor node.

It is also possible to send control messages such as location or receive status inquiries from the control center to a specific sensor node. Again, since there does not necessarily exist an active connection to the sensor node messages need to be queued. Hence, the applications in the control center are responsible for implementing proper queuing and retry mechanisms.

Security and message integrity are critical aspects of the overall monitoring system. If thieves were able to tamper with the message contents then they could easily spoof the system. Security is essential and needs to be implemented in each application that sends or receives messages as part of the monitoring system. This brings up another issue, scalability. Implementing asynchronous communication and security components for each application in a small system may work for experiments but is not feasible for large production environments. In terms of the cargo monitoring scenario for trains there could be many control centers, thousands of sensor

nodes and even more sensors on containers. This is a very common scenario for sensor network deployments even though the particular details of the deployments may be different. In this paper we demonstrate that by using JMS in these mobile monitoring environments it is possible to overcome the common problems discussed above. In particular we focus on the problems of asynchronous communication, message security and integrity, and scalability.

A. Asynchronous Communication

Reliable communication between control centers and the sensor networks cannot always be ensured. Additionally communication is based on the form of underlying connectivity that is provided. The connectivity may vary. The system could use a 3G system when available and resort to the satellite communication only when needed; exposing several issues.

First, message sizes should be small in order to accommodate for the slow speeds such as satellite communication when needed. Possible optimizations are discussed in III-B4 but compression or conversion into binary formats are suitable options here.

Second, in order to address reliable transmission of messages either a *store-and-forward* or a *resend* mechanism needs to be implemented on both communication ends. The *store-and-forward* technique in this context would mean that the sensor networks need to hold to the data they capture until connectivity is established. By contrast, in the *resend* scenario they would attempt to transmit the data continuously or with a backoff timer.

B. Message Security and Integrity

The data that is produced by sensor networks will likely be sensitive and needs to be kept private. This is especially true for systems whose main purpose is to provide monitoring of cargo. Cargo information as well as status updates and events should only be visible to authorized entities. Furthermore it is critical that messages being transmitted cannot be tampered with, for example control messages that allow the opening of cargo containers.

In this sense it is also important to distinguish between *point-to-point* and *end-to-end* security. Using transit networks or message relay mechanisms is not possible when messages are secured in a point-to-point manner because security may be compromised at each individual connection point. However, in *end-to-end* system security it is possible for messages to pass through individual connection points. Another issue that always needs to be kept in mind is that while control centers often have adequate storage and computing power individual sensors or sensor networks may not. This can be a challenge when implementing security for the targeted scenarios.

C. Scalability

Sensor networks in general can be set up in two basic ways. First, after an initial configuration they repeatedly report their sensor data to a control center. Second, a control center sends out messages to the sensors or sensor networks in order to

control their reporting or inquire for specific sensor data. Thus efficient management and scalability can become an issue.

Even though the most common scenario is running a single setup with one central control center or base station and multiple sensors or sensor networks connecting to it, the integration of multiple systems can be problematic. There are issues in dealing with multiple control centers and multiple sensor networks that need to be explored. This is especially important when it comes to managing policies and subscriptions properly.

III. RELATED WORK

A. Java Message Service

Musolesi et al. [1] present their experiences in implementing a system called EMMA (Epidemic Messaging Middleware for Ad hoc networks) based on JMS. In particular they identified the need to adapt JMS in order to be applicable for mobile ad hoc networks. Their approach consists in synchronization of queues using a middleware layer that also manages reachability of individual nodes. For message delivery in partially connected networks they make use of an approach called epidemic routing which is described by [2] which works by propagating messages to neighbors, their neighbors and so on. In contrast the solution discussed here and implemented in the TSSN is standards based using the original JMS specification and therefore more compatible with other systems.

Vollset et al. [3] present a middleware platform built for mobile ad-hoc networks. Their solution is “serverless” in the sense that after an initial setup all the participating entities have a local copy of the JMS configuration. Furthermore they implement a new multicast protocol for delivering messages on JMS *topics* to their *subscribers*. Their platform again is an adaptation of the original JMS standard whereas the solution presented here makes use of a specified and standardized implementation and shows that JMS can be used unaltered.

In general the Java Message Service is primarily used in always connected systems such as the one described by [4]. The Mission Data Processing and Control Subsystem (MPCS) [4] utilizes JMS for different levels of event notifications. However, the communication link with the flight systems is custom. Although an extreme case, it seems that using the Java Message Service for establishing mobile connectivity is undervalued. Another, more realistic example is the Remote Real-time Oil Well Monitoring System [5] where clients receive event notifications via JMS but the data that is collected by the remote terminal units is sent to the data processing station using a custom process.

B. Web Services

Service Oriented Architectures (SOA) present a flexible approach to some of the problems mentioned earlier such as message security and scalability. The idea is to implement specific functionality in web services that communicate with each other using standardized interfaces. Message exchanges in general use the flexible SOAP message format [6]. This has a number of advantages as for instance routing and security are available as extensions to it. JMS is able to transmit SOAP

messages. Hence, applying JMS enables the use of web service specifications in mobile monitoring environments.

1) *WS-Addressing*: The *WS-Addressing* core specification by [7] and its SOAP binding by [8] defines how message propagation can be achieved using the SOAP message format. Usually the transport of messages is handled by the underlying transport protocol but there are several advantages of storing this transport information as part of the header in the actual SOAP message. For example, it allows the routing of messages across different protocols and management of individual flows and processes within web services.

The Java Message Service uses a similar concept for its addressing but its properties are adapted to the management of messages in queues. However, since SOAP messages can be transported over JMS flexible routing of messages is preserved.

2) *WS-Security*: The *WS-Security* specification as described by [9] deals with the many features needed to achieve so-called *end-to-end* message security. This provides security throughout message routing and overcomes the limitations of so-called *point-to-point transport layer security* such as HTTPS. Furthermore, the specification aims to provide support for a variety security token formats, trust domains, signature formats and encryption technologies.

Whenever SOAP messages are transported using the Java Message Service, *WS-Security* can be applied. In this scenario JMS simply acts as a tunnel.

3) *WS-ReliableMessaging*: Without additional specifications like *WS-ReliableMessaging* [10] the delivery of SOAP messages is based purely on best effort and cannot necessarily be guaranteed. The Java Message Service provides several mechanisms for dealing with message reliability issues. Within *transactions* messages are *acknowledged* and if necessary *redelivered*. When a message carries the *persistent* attribute, JMS message *brokers* store the message in order to be able to recover it in case of a failure.

4) *Efficient Data Transmission*: The SOAP 1.2 Primer [6] includes references to several enhancements of the original SOAP standard. In particular they deal with potential performance problems and the need for binary data transport in SOAP. The *XML-binary Optimized Packaging (XOP)* specification [11] defines the use of *MIME Multipart/Related* messages provided by [12] to avoid encoding overhead that occurs when binary data is used directly within the SOAP message. XOP extracts the binary content and uses URIs to reference it in the so-called *extended part* of the message. An abstract specification that uses this idea is the *Message Transmission Optimization Mechanism (MTOM)* [13].

Another extension of the SOAP standard is the *Resource Representation SOAP Header Block (RRSHB)* [14] that allows for caching of data elements using so-called *Representation header blocks*. They contain resources that are referenced in the SOAP *Body* which might be hard to retrieve or simply referenced multiple times. Instead of having to reacquire them over and over again, a service may choose to use the cached objects which speeds up the overall processing time.

ActiveMQ, which is the JMS implementation that is used in the Transportation Security SensorNet, allows several different protocols (e.g. AMQP [15], OpenWire [16], REST [17], Stomp

[18], XMPP [19]), to be used for message transmission. By default it uses the OpenWire protocol, an optimized binary format for fast and efficient communication.

IV. PROPOSED SOLUTION

In order to be flexible and provide a suitable solution for a mobile monitoring environment a transparent *store-and-forward* approach is used here. A *resend* mechanism is often implemented directly in the application which makes it inflexible. However, the *store-and-forward* approach allows for a more efficient and scalable centralized storage pool that is automatically forwarding the messages.

How JMS can be applied effectively is described here using the Transportation Security SensorNet (TSSN) [20] as an example. The TSSN provides monitoring capabilities in mobile environments and makes use of JMS. The TSSN uses a SOA approach for monitoring cargo in motion along trusted corridors. The system is built using web service specifications and utilizes a Java Message Service implementation for connectivity between its Virtual Network Operation Center (VNO), the control center in this case, and the Mobile Rail Networks (MRN), which contains the sensor nodes and sensors, it monitors.

The TSSN uses the Java Message Service through one of its open-source implementations called ActiveMQ which is described in detail by Snyder et al. [21]. Each application in the TSSN is a web service. These web services can be utilized through their JMS addresses. ActiveMQ establishes a so-called *queue* for each web service and uses these *queues* to *store-and-forward* messages to them.

This *queue* approach has the advantage that applications do not need to be modified and implement their own *store-and-forward* or *resend* mechanisms. It is also transparent to clients of the web services since apart from using another address, the JMS address, interfacing with the web services stays the same.

In this paper we explain the basic concepts of JMS and in particular how they relate to mobile monitoring environments and one implementation, the TSSN. Furthermore the details for the JMS implementation within the TSSN are discussed.

A. Java Message Service

The Java Message Service [22] provides a standardized specification for synchronously and asynchronously transporting messages using queues. Its implementation is vendor specific but the interfaces are clearly defined in the specification so that in theory this is an open system where changing vendors is possible. The following sections describe the Java Message Service in detail.

1) *Components*: In the JMS context clients are called *producers* when they create and send messages. The receiving end is called a *consumer*. Note that a client can be both, a producer and a consumer, at the same time. Clients connect to JMS *providers* which are entities that have the specified interfaces to send and receive messages.

Since most of the connections in general are point-to-point, a so-called *queue* is the most commonly used *destination* of a message. It contains messages from *producers* to a

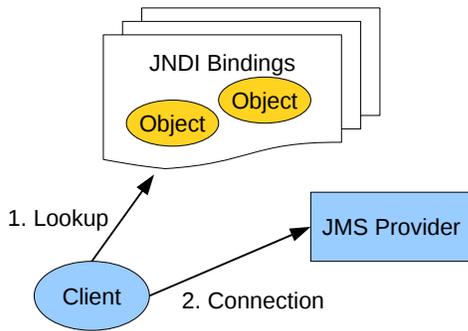


Fig. 1. JMS administration according to [22]

single *consumer* that have not been received. Within the TSSN unique *queues* are used to represent the individual web services. Messages are usually delivered in order *First In, First Out* (FIFO) following the basic principle of a queue but this is dependent on the underlying implementation of JMS.

Topics have multiple *consumers* and can have one or many *producers* publishing messages. They are used in publish-subscribe models and contain messages that have not yet been published.

A *message* can be any object or data that needs to be transported using JMS. The Java Message Service describes messages as entities that consist of a *header* which contains identification and routing information and a body carrying the data. Additional properties such as application, provider or standards specific properties can be attached to messages. This is effectively used in providing things like security or reliable messaging.

Note that since JMS per se does not define a message format, implementation may vary significantly. For service oriented architectures the agreed standard message format is SOAP. Easton et al. [23] describe in detail how SOAP can be used within the Java Message Service. Because the TSSN is based on SOA and uses SOAP messages it is able use web service specifications as part of JMS and therefore provide features such as *WS-Addressing* and *WS-Security* as described in III-B.

2) *Java Naming and Directory Interface*: In order to identify objects within the Java Message Service implementation in a standardized way the specification makes use of the Java Naming and Directory Interface (JNDI) Application Programming Interface [24]. JNDI provides a directory service for objects. This process is used for so-called *connection factories* which are used to establish connections and *destinations* which are either *queues* or *topics* in order to increase portability and ease of administration. JMS clients look up objects and use them in connections as shown in Figure 1. TSSN uses local JNDI repositories for JMS lookups and a combination of *hostname* and *web service name* for uniquely naming *queues*.

3) *Messaging Models*: JMS supports the two common messaging models; point-to-point and publish-subscribe. These are also called message *domains*. Both of them allow for true asynchronous communication in which the message *consumer* does not need to be connected to the *producer* at the time



Fig. 2. Point-to-Point messaging

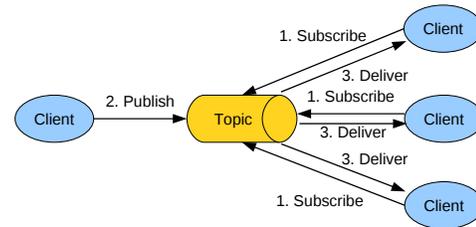


Fig. 3. Publish/Subscribe messaging

when the message is sent and vice versa.

a) *Point-to-point*: This messaging model makes use of queues and is shown in Figure 2. Its main application is a request-response type of message exchange. Messages in this model are truly unique in the sense that once the *consumer* received and acknowledged the message it is removed from the queue. While there can be only a single *consumer*, messages can be put on the queue by multiple *producers*. This is the model used in the TSSN because message propagation throughout the mobile monitoring system is done from one web service to another.

b) *Publish-subscribe*: Whenever there is the need for multiple *consumers* to receive messages, a subscription model is useful. The *consumers* subscribe to a specific *topic* and receive messages as soon as they are published by one or multiple *producers* as shown in Figure 3. There exists no direct connection between *publishers* and *subscribers*.

Two types of subscriptions are possible in this model. The *subscriber* is either continuously connected to the *topic* and checks for new publications or a *durable subscription* is created in which the messages are kept within the topic while the *subscriber* is not connected. Upon reconnection messages are delivered to the *subscriber*.

In the TSSN the JMS *publish-subscribe* messaging model is currently not used since publications are handled using the web service standard *WS-Eventing* for SOAP messages. However, JMS *publish-subscribe* presents a flexible approach to scalability. Since switching between messaging models is only based on configuration parameters and not on a different implementation it is possible to use JMS *publish-subscribe* effectively in mobile monitoring environments as well.

B. Transportation Security SensorNet

The Transportation Security SensorNet [20] as shown in Figure 4 uses a Service Oriented Architecture approach for monitoring cargo in motion along trusted corridors. The complete system provides a web services based sensor management and event notification infrastructure that is built using open standards and specifications. Particular functionality within the system has been implemented in web services that provide interfaces according to their respective web service

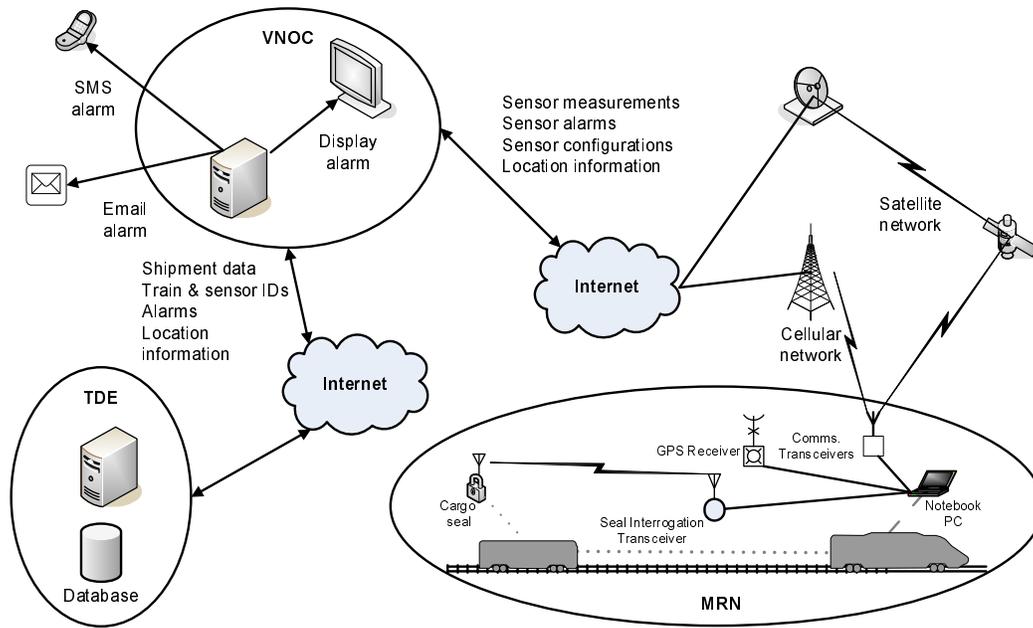


Fig. 4. TSSN physical architecture adapted from [25]

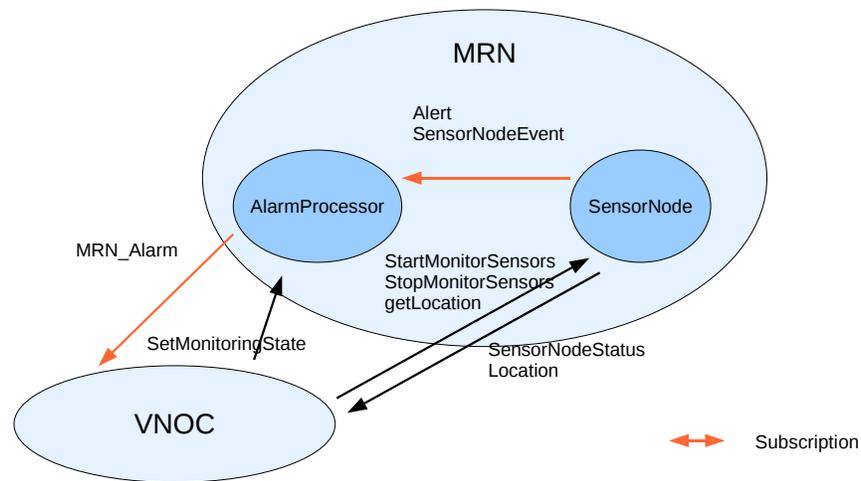


Fig. 5. Mobile Rail Network message overview from [20]

specifications. This web services based implementation allows for platform and programming language independence and offers compatibility and interoperability with other systems.

The TSSN represents the integration of SOA, Open Geospatial Consortium (OGC) specifications and sensor networks. Previous systems and research focused either on the combination of SOA and OGC specifications or on OGC standards and sensor networks. However, the TSSN shows that all three can be combined and that this combination provides capabilities to the transportation and other industries that have not existed before. In particular, the preminent lack of performance in mobile sensor network environments has previously limited the application of web services because they have been perceived as too slow and producing a lot of overhead. The TSSN, as

shown by the results in [20], demonstrates that with proper architecture and design the performance requirements of the targeted scenario can be satisfied.

Furthermore, unlike existing proprietary implementations the Transportation Security SensorNet allows sensor networks to be utilized in a standardized and open way through web services. Sensor networks and their particular communication models led to the implementation of asynchronous message transports in SOA and are supported by the TSSN.

Within the TSSN the Mobile Rail Network (MRN), which is shown in Figure 5, represents a train-mounted sensor network (sensors and sensor node) that monitors seals on cargo containers. The MRN is able to receive control messages such as when to start and stop monitoring. When an event is

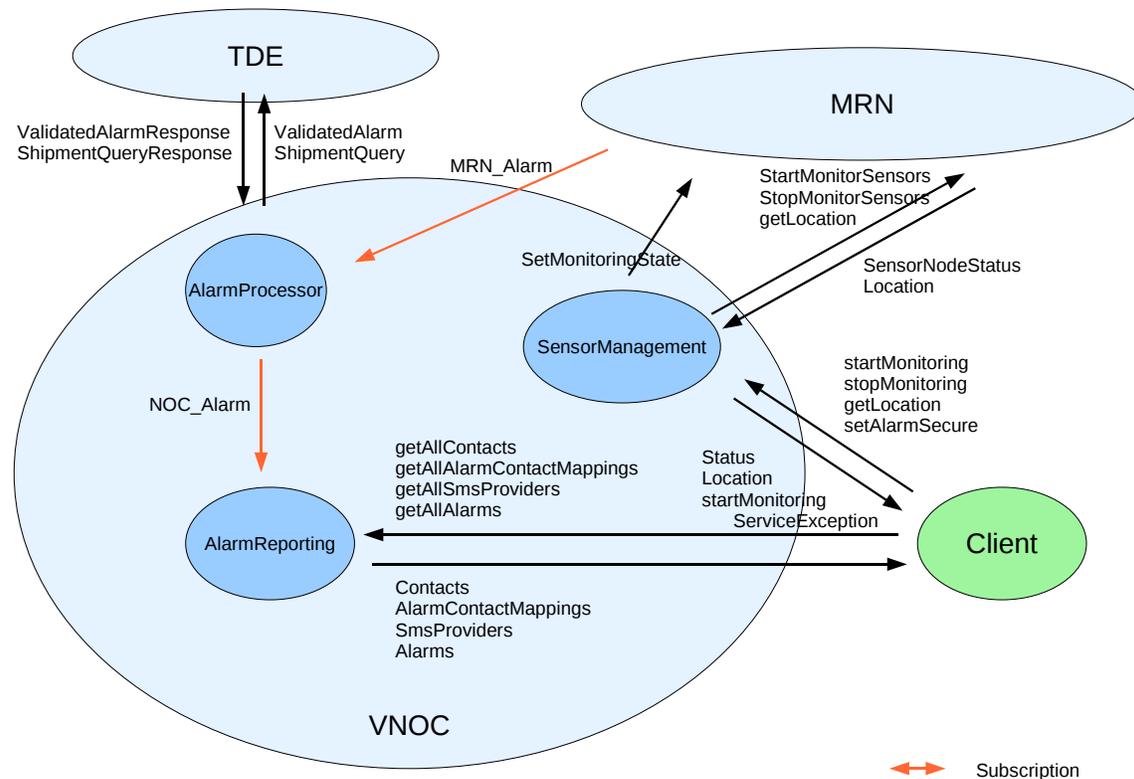


Fig. 6. Virtual Network Operation Center message overview from [20]

detected it is transmitted from a sensor (seal) to the sensor node where it is analyzed to determine whether or not to send out a notification to the VNOc (Figure 6). Sensor management and correlation of events with shipment and route information is then performed in the VNOc. According to specified mappings, people and organizations that subscribed to these notifications then receive emails and/or sms messages containing detailed information about the nature of the event.

In terms of the communication the critical link is between the Mobile Rail Network and the Virtual Network Operation Center because it cannot be guaranteed that there always exists a link and hence an asynchronous communication model had to be implemented. An approach that is able to deal with message queuing on both ends of the communication is the integration of the Java Message Service as the transport. The TSSN implementation fully supports asynchronous communication using the so-called *Enterprise Service Bus* queues in order to send and receive messages.

1) *Axis2*: The TSSN is based on the Apache Axis2 web services software stack. By default Axis2 uses request-response in a *synchronous* manner. This means that the client has to wait and is therefore *blocking* until it receives the response from the service. In certain scenarios, for instance when the service needs a large amount of processing time, the client can experience timeouts. Furthermore, in the TSSN where the MRN is only intermittently connected to the VNOc, *synchronous* communication is not feasible.

A better option is to make the communication between

services *asynchronous*. This resolves timeout issues and deals with connections that are only temporary. The following aspects were taken into consideration when developing the *asynchronous* communication for this case:

a) *Client*: The client needs to make changes from *synchronous* to *asynchronous* messaging in regard to how the request is sent out. Axis2 provides a low-level *non-blocking client API* and additional methods in the service stubs that allow callbacks to be registered. These so-called *AxisCallbacks* need to implement two methods, one that is being invoked whenever the response arrives and the other to define what happens in case of an error.

b) *Transport Level*: Depending on the transport protocol that is being used, Axis2 supports the following approaches.

- *One-way* uses one channel for the request and another one for the response such as used in the Simple Mail Transfer Protocol (SMTP)
- *Two-way* allows the same channel to be used for the request and the response, for example HTTP

For asynchronous communication to work in the TSSN the two-way approach was modified as part of [20] through the Axis2 *client API* which provides the option of using a *separate listener*. This tells the service that it is supposed to use a new channel for the response. In order to correlate request and response messages Axis2 makes use of the *WS-Addressing* specification, in particular the *RelatesTo* field.

c) *Service*: The final piece of asynchronous communication is to make the service processing asynchronous as well.

```

<transportReceiver name="jms"
class="org.apache.axis2.transport.jms.JMSListener">
<parameter name="myTopicConnectionFactory">
<parameter name="java.naming.factory.initial">
org.apache.activemq.jndi.ActiveMQInitialContextFactory
</parameter>
<parameter name="java.naming.provider.url">
tcp://localhost:61616</parameter>
<parameter name="transport.jms.ConnectionFactoryJNDIName">
TopicConnectionFactory</parameter>
</parameter>

<parameter name="myQueueConnectionFactory">
<parameter name="java.naming.factory.initial">
org.apache.activemq.jndi.ActiveMQInitialContextFactory
</parameter>
<parameter name="java.naming.provider.url">
tcp://localhost:61616</parameter>
<parameter name="transport.jms.ConnectionFactoryJNDIName">
QueueConnectionFactory</parameter>
</parameter>

<parameter name="default">
<parameter name="java.naming.factory.initial">
org.apache.activemq.jndi.ActiveMQInitialContextFactory
</parameter>
<parameter name="java.naming.provider.url">
tcp://localhost:61616</parameter>
<parameter name="transport.jms.ConnectionFactoryJNDIName">
QueueConnectionFactory</parameter>
</parameter>
</transportReceiver>

```

Fig. 7. JMS transport receiver configuration in axis2.xml

```

<transportSender name="jms"
class="org.apache.axis2.transport.jms.JMSSender" />

```

Fig. 8. JMS transport sender configuration in axis2.xml

This is done by specifying so-called *asynchronous message receivers* in the services configuration in addition to the *synchronous* ones.

Axis2 then uses the *ReplyTo* field of the *WS-Addressing* header in the client as a sign to send an immediate *acknowledge* of the request back to it. Furthermore it processes the request in a new thread and sends the response out when it is done, allowing the communication to be performed in asynchronous manner completely.

There exist various forms of transport protocols that are suitable for *asynchronous* communication. Axis2 by default supports HTTP, SMTP, and JMS as *asynchronous* transports but other transports can easily be defined and plugged in.

In order to allow for the TSSN to use JMS as a transport the following items were added to the Axis2 configuration by the authors. First, a so-called *transport receiver* for JMS as shown in Figure 7. This represents the receiving end of the communication and allows web services and clients to consume JMS messages by creating a JMS address for them. In particular, connection factories are set up for *queues* and *topics*. Second, a *transport sender* shown in Figure 8 allows JMS messages to be produced.

Axis2 by default sets up a *queue* for each of the services and uses the service name as the *queue* name. Since a service is not necessarily unique this name can be changed in the service configuration (Figure 9). For the Mobile Rail Network this naming consists of the node id which is used to represent a sensor network and the name of the service. For the Virtual Network Operation Center the name is made up of the host

```

<parameter name="transport.jms.ConnectionFactory">
myQueueConnectionFactory</parameter>
<parameter name="transport.jms.Destination">
TSSN_NODE/2222/MRN_SensorNode</parameter>

```

Fig. 9. JMS service queue name configuration in services.xml

```

<broker xmlns="http://activemq.apache.org/schema/core"
brokerName="mrn2222"
dataDirectory="\${activemq.base}/data">
...
</broker>

```

Fig. 10. ActiveMQ broker configuration in activemq.xml

```

<transportConnectors>
<transportConnector name="openwire"
uri="tcp://localhost:61616" />
<transportConnector name="ssl"
uri="ssl://localhost:61617"/>
</transportConnectors>

```

Fig. 11. ActiveMQ transport connector configuration in activemq.xml

on which the service is run and its name. This makes it possible to easily identify queues and avoid misconfiguration of ActiveMQ while also offering a naming scheme that is scalable.

2) *ActiveMQ*: Apache ActiveMQ is an open source implementation of the Java Message Service and is used by the TSSN for JMS messaging. A detailed introduction is given by Snyder et al. [21]. It is a JMS message broker mostly used in enterprise systems where high bandwidth connectivity is a given and throughput is most important. Note that the version used within the TSSN had to be modified by the authors because ActiveMQ could not work correctly without an existing and permanent Internet connection. However, being able to function without constant connectivity is essential in mobile monitoring environments. For example, the connection between the Virtual Network Operation Center and the Mobile Rail Networks in the TSSN may be intermittent.

The following sections explain the important components of ActiveMQ that are used in the TSSN and provide configuration details.

a) *Broker*: A *broker* is responsible for managing queues and topics. It receives message from *producers* which connect to it and delivers them to the according *consumers*. The configuration for a broker at a Mobile Rail Network is shown in Figure 10.

b) *Transport Connectors*: *Brokers* allow *producers* and *consumers* to use various protocols to connect to it. In ActiveMQ these connectivity entities are defined as *transport connectors*. The TSSN configures the services and clients use TCP in order to connect to the *broker* (Figure 11). Another use of the specified protocols is for inter-broker communication which is explained in detail later.

c) *Network Connectors*: Multiple *brokers* can form a *network of brokers* using *network connectors*. This is allows the use of *distributed queues* and is the setup that is used to connect Virtual Network Operation Center and Mobile Rail Networks. In order to be flexible the configuration of a so-called *network bridge* is initiated by the Mobile Rail Networks (Figure 12). Establishing a *duplex connection* then enables

```

<networkConnectors>
<networkConnector
name="MRN2222network"
uri="static://(tcp://laredo.ittc.ku.edu:61616)?
initialReconnectDelay=5000&
useExponentialBackOff=false"
duplex="true"
dynamicOnly="false"
networkTTL="5"/>
</networkConnectors>

```

Fig. 12. ActiveMQ MRN network connector configuration in activemq.xml

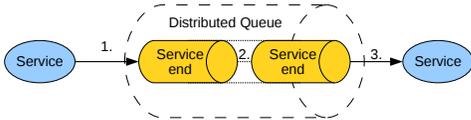


Fig. 13. One-way JMS message transmission

messages to be forwarded in both directions. The advantage here is that the VNOC does not need to be reconfigured every time a new MRN is set up.

ActiveMQ makes use of the OpenWire protocol [16] which is an optimized binary compressed format tailored to efficient management of JMS *queues* and *topics* as well as network connectivity. This is another advantage of using ActiveMQ since it makes sure that communication between brokers is bandwidth efficient which is essential for slow and unstable connections such as satellite links.

d) *Distributed Queues*: Connections from the VNOC to the MRN and vice versa are *point-to-point* which corresponds to *queues* in the Java Message Service. *Queues* can be distributed across several *brokers*. Whenever the *brokers* are connected to each other they exchange information about which *broker* has the *consumer* and the other *brokers* forward their queue messages to that *broker*. The two common types of message exchanges are explained in the following paragraphs.

Notification messages require only one-way communication as shown in Figure 13. Within the TSSN a web service acts as a *producer* and puts the notification onto the queue which corresponds to the web service it wants to notify. This is done by using the specified *transport connector* to connect to the local *broker* and deliver the message to it. The *broker* then puts the message on the *queue end* that it manages. Whenever the *broker* can contact the *queue end* with the consumer it forwards the message. The receiving web service uses a listener to detect when its *queue* at the broker contains new messages. It then uses its local *transport connector* to consume the notification.

Control messages that are sent by the VNOC to the MRN are good examples of two-way communications. As shown in Figure 14 in a request-response scenario the client creates a *temporary queue* at its local broker that only itself knows about. This is where the response message will be put. The request then follows the usual path from the local *transport connector* to the local *broker*, from the local *broker* to the *broker* with the specified *consumer* and then using the remote *transport connector* to the according web service. The JMS message that is transmitted contains a *ReplyTo* field with the

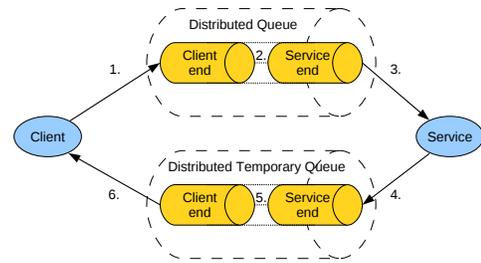


Fig. 14. Two-way JMS message transmission

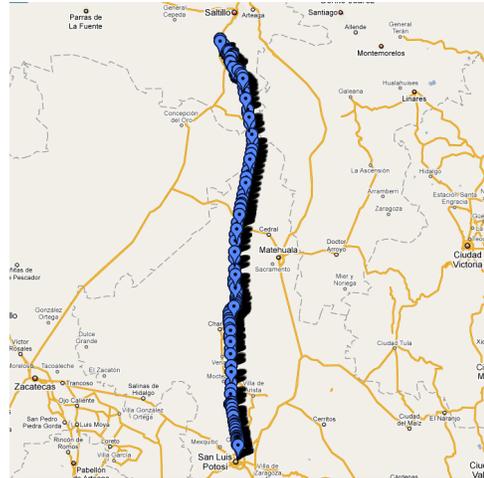


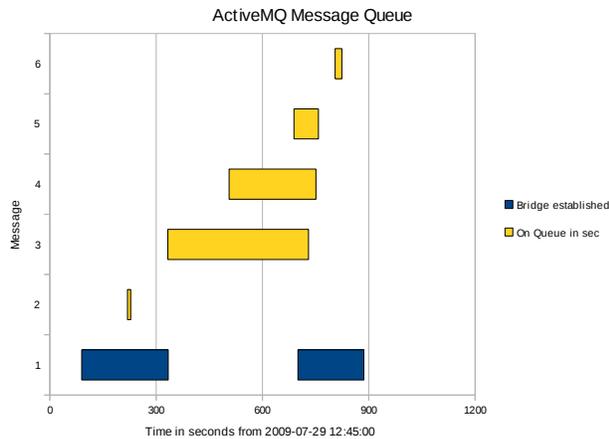
Fig. 15. Route for longhaul field trial, route starts at San Luis Potosi and ends approximately 210 miles down the track

temporary queue that is used for the response. The response is then sent to back using the web service's local *transport connector*, local *broker*, remote *broker* until it is consumed from the *temporary queue* by the original client.

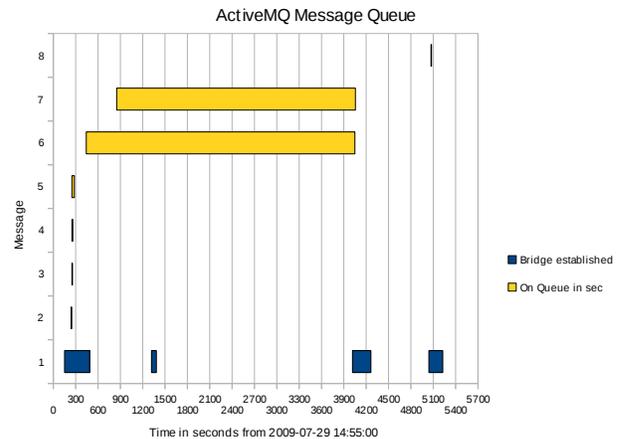
In the TSSN all of the queue creation, message queuing and brokering is transparent to the web services. Whenever asynchronous communication using the Java Message Service is required the clients and web services simply use JMS addresses instead of the default HTTP ones which can be set in their configuration files. This makes the solution very scalable and flexible since a *store-and-forward* mechanism does not need to be implemented in each web service but is provided by an ActiveMQ JMS message broker. This holds true not only for Service Oriented Architectures that make use of web services but is also applicable to other systems.

V. RESULTS

The described system above has been successfully tested in a field trial. Here are results from two scenarios which were explored: a stationary scenario and a mobile scenario, carried out by mounting the equipment onto a train. Throughout the experiments communication between the VNOC and the MRN in the TSSN is using a satellite link, in this case Iridium at 2.4 kb/s. Both scenarios were performed as part of a longhaul trial in Mexico (see Figure 15).



(a) Initial test



(b) Follow-up tests

Fig. 16. ActiveMQ message queue for stationary scenario

A. Stationary

Figures 16a and 16b show tests results acquired from when the MRN was set up in a rail yard but not mounted on a train and not moving. The time each message spent on the distributed queues as well as when the MRN and the VNOC brokers had a JMS network bridge established and were therefore fully connected is displayed. It can be seen that while all messages were successfully transmitted the time a message was on a queue is dependent on the quality of the satellite connection. Only once the satellite connection is stable enough for ActiveMQ to establish a network bridge messages can be transmitted.

B. Mobile

The more interesting scenario is to use TSSN as a mobile monitoring environment. For this purpose the MRN was deployed on a train and sensors attached to cargo containers. Figure 17 shows results of roughly the first hour (8:30-9:30am July 30th 2009) of the longhaul trial along the path shown in Figure 15. Due too a hardware problem after about 9:30am the system clock synchronization is significantly off and the remaining data, especially time measurements, cannot be analyzed. Therefore, only the time before the hardware issue occurred is shown in Figure 17. However, it is important to note that the TSSN kept operating correctly for more than 32 hours and ActiveMQ was successfully transmitting and receiving messages.

Figure 17 shows in detail when messages were put on a queue, when they were consumed and at which time the JMS network bridge, actual connectivity, was established. A comparison of times message required to be transmitted from the MRN to the VNOC is shown in Table I. Whenever connectivity, in ActiveMQ a so-called bridge, is established the actual message transmission takes about 11.6 seconds on average. This is in stark contrast to when the satellite link is down and needs to be established before sending out messages. In that case it took 616.2 seconds on average with

TABLE I
ELAPSED TIME FROM MRN TO VNOC DURING TRIAL IN SECONDS

	Minimum	Maximum	Mean	Median	Std. Dev
Link Down	31.29	1273.1	616.26	553.23	411.36
Link Up	5.85	40.53	11.62	6.02	10.77
Average Case	5.85	1273.1	481.90	430.97	441.86

TABLE II
COMPARISON OF ELAPSED TIME FROM MRN TO VNOC OF LONGHAUL TO SHORThAUL TRIAL IN SECONDS

	Minimum	Maximum	Mean	Median	Std. Dev
Shorthaul	0.45	2.90	1.89	1.94	0.62
Longhaul	5.85	1273.1	481.90	430.97	441.86

the slowest message being received 1273.1 seconds or more than 21 minutes after it was sent.

The key characteristic here is the availability of the satellite link. The trial was performed in a mountainous environment where the satellite view was partially obstructed and hence the times measured may not be the same in a different geographic region. Looking at the average case of about 7 minutes per message transmission though the system is found to be in range of mobile monitoring environments.

A comparison of these results to a previous shorthaul trial described in [20] is shown in Table II. During the shorthaul the MRN was continuously connected to the VNOC using a GSM modem with a peak throughput of about 700 kb/s. Looking at the minimum times and assuming this as the best case scenario the satellite configuration is slower by a factor of about 13.

VI. CONCLUSION

As discussed in the previous sections the approach of using JMS in mobile monitoring environments works. We showed that Java Message Service technology can be utilized to provide drop-in connectivity between disparate and delay tolerant systems. Previously JMS was primarily used in always connected high bandwidth scenarios but its concepts and

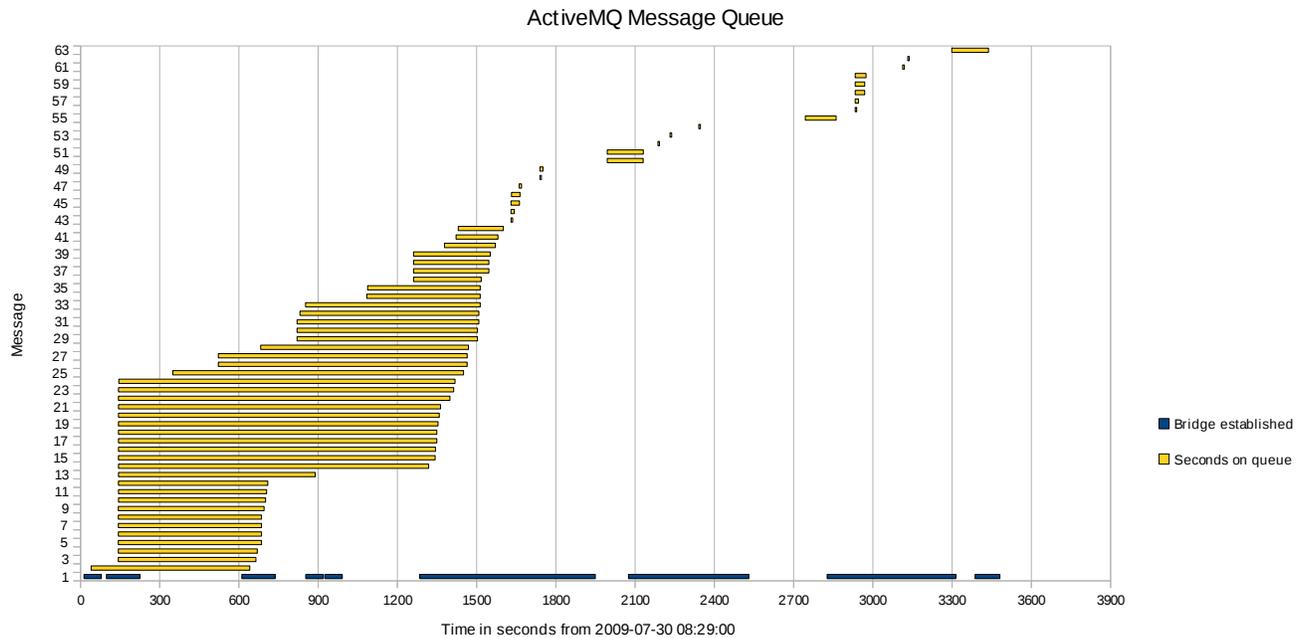


Fig. 17. ActiveMQ message queue for mobile scenario

techniques are useful in mobile monitoring environments as well.

JMS provides a transparent asynchronous communication model that is scalable and flexible since a *store-and-forward* mechanism does not need to be implemented in each component but is provided by a JMS message broker. This is done using distributed queues that are managed by network connectors as described above.

Since JMS allows the transport of all types of messages including SOAP messages, web service specifications were used here to provide features such as *end-to-end* message security and integrity. In terms of scalability JMS makes it possible to connect disparate systems with limited effort without having to implement *store-and-forward*, *resend* mechanisms and security again and again. Furthermore, JNDI and support for different messaging models enhance scalability for JMS based systems.

In this paper we presented a new and flexible approach to deal with challenges such as intermittent and low-bandwidth communication in mobile monitoring environments. This approach is to utilize the features that the Java Message Service provides to address the issues of asynchronous communication, message security and integrity, and scalability. We have shown that this is possible and presented an implementation of a mobile monitoring system called TSSN that successfully uses the approach.

ACKNOWLEDGMENT

This work was supported in part by Oak Ridge National Laboratory (ORNL) Award Number 4000043403. This material is also partially based upon work supported while V. S. Frost was serving at the National Science Foundation.

REFERENCES

- [1] M. Musolesi, C. Mascolo, and S. Hailes, "Adapting asynchronous messaging middleware to ad hoc networking," in *MPAC '04: Proceedings of the 2nd workshop on Middleware for pervasive and ad-hoc computing*. New York, NY, USA: ACM, 2004, pp. 121–126.
- [2] A. Vahdat and D. Becker, "Epidemic Routing for Partially-Connected Ad Hoc Networks," Duke University, Tech. Rep., 2000.
- [3] E. Vollset, D. Ingham, and P. Ezhilchelvan, "JMS on Mobile Ad-Hoc Networks," in *In Personal Wireless Communications (PWC)*. Springer-Verlag, 2003, pp. 40–52.
- [4] D. Allard, "Development of a ground data messaging infrastructure for the mars science laboratory and beyond," in *Aerospace Conference, 2007 IEEE*, March 2007, pp. 1–8.
- [5] L. Hongsheng, W. Yu, D. Yongzhong, and P. Zhongxiao, "Implementation of network-computing and nn based remote real-time oil well monitoring system," in *Neural Networks and Brain, 2005. ICNN&B '05. International Conference on*, vol. 3, Oct. 2005, pp. 1810–1814.
- [6] Y. Lafon and N. Mitra, "SOAP version 1.2 part 0: Primer (second edition)," W3C, W3C Recommendation, Apr. 2007, <http://www.w3.org/TR/2007/REC-soap12-part0-20070427/>.
- [7] M. Gudgin, M. Hadley, and T. Rogers, "Web services addressing 1.0 - core," W3C, W3C Recommendation, May 2006, <http://www.w3.org/TR/2006/REC-ws-addr-core-20060509/>.
- [8] M. Gudgin, M. Gudgin, M. Hadley, T. Rogers, T. Rogers, and M. Hadley, "Web services addressing 1.0 - SOAP binding," W3C, W3C Recommendation, May 2006, <http://www.w3.org/TR/2006/REC-ws-addr-soap-20060509/>.
- [9] K. Lawrence, C. Kaler, A. Nadalin, R. Monzillo, and P. Hallam-Baker, "Web Services Security: SOAP Message Security 1.1 (WS-Security 2004)," OASIS, OASIS Standard, Feb. 2006, <http://docs.oasis-open.org/wss/v1.1/wss-v1.1-spec-os-SOAPMessageSecurity.pdf>.
- [10] P. Fremantle, S. Patil, D. Davis, A. Karmarkar, G. Pilz, S. Winkler, and mit Yalinalp, "Web Services Reliable Messaging (WS-ReliableMessaging) Version 1.1," OASIS, OASIS Standard, Jun. 2007, <http://docs.oasis-open.org/ws-rx/wsrml/200702/wsrml-1.1-spec-os-01.pdf>.
- [11] N. Mendelsohn, H. Ruellan, M. Gudgin, and M. Nottingham, "XML-binary optimized packaging," W3C, W3C Recommendation, Jan. 2005, <http://www.w3.org/TR/2005/REC-xop10-20050125/>.
- [12] E. Levinson, "The MIME Multipart/Related Content-type," RFC 2387 (Proposed Standard), Aug. 1998. [Online]. Available: <http://www.ietf.org/rfc/rfc2387.txt>

- [13] M. Nottingham, H. Ruellan, N. Mendelsohn, and M. Gudgin, "SOAP message transmission optimization mechanism," W3C, W3C Recommendation, Jan. 2005, <http://www.w3.org/TR/2005/REC-soap12-mtom-20050125/>.
- [14] M. Gudgin, Y. Lafon, and A. Karmarkar, "Resource representation SOAP header block," W3C, W3C Recommendation, Jan. 2005, <http://www.w3.org/TR/2005/REC-soap12-rep-20050125/>.
- [15] "Advanced Message Queuing Protocol (AMQP) version 0-10 Specification," Specification, 2009, <http://www.amqp.org>.
- [16] "OpenWire Version 2 Specification," Apache ActiveMQ, Specification, 2009, <http://activemq.apache.org/openwire-version-2-specification.html>.
- [17] R. T. Fielding, "Architectural styles and the design of network-based software architectures," Ph.D. dissertation, University of California, Irvine, 2000.
- [18] J. Strachan, "Stomp Protocol Specification, Version 1.0," FuseSource, Specification, 2005, <http://stomp.codehaus.org/Protocol>.
- [19] P. Saint-Andre, "Extensible Messaging and Presence Protocol (XMPP): Core," RFC 3920 (Proposed Standard), Internet Engineering Task Force, Oct. 2004. [Online]. Available: <http://www.ietf.org/rfc/rfc3920.txt>
- [20] M. Kuehnhausen, "Service Oriented Architecture for Monitoring Cargo in Motion Along Trusted Corridors," Master's thesis, University of Kansas, Jul. 2009.
- [21] B. Snyder, D. Bosanac, and R. Davies, *ActiveMQ in Action*. Manning Publications, 2009.
- [22] M. Hapner, R. Burridge, R. Sharma, J. Fialli, and K. Stout, "Java(TM) Message Service Specification," Sun Microsystems, Specification, 2002, <http://java.sun.com/products/jms/>.
- [23] P. Easton, B. Mehta, and R. Merrick, "SOAP over java message service 1.0," W3C, W3C Working Draft, Jul. 2008, <http://www.w3.org/TR/2008/WD-soapjms-20080723>.
- [24] "Java Naming and Directory Interface Application Programming Interface (JNDI API)," Sun Microsystems, Specification, 1999, <http://java.sun.com/products/jndi/>.
- [25] D. T. Fokum, V. S. Frost, D. DePardo, M. Kuehnhausen, A. N. Oguna, L. S. Searl, E. Komp, M. Zeets, J. B. Evans, and G. J. Minden, "Experiences from a Transportation Security Sensor Network Field Trial," Information Telecommunication and Technology Center, University of Kansas, Lawrence, KS, Tech. Rep. ITTC-FY2009-TR-41420-11, June 2009.



Technical Report

Implementing Web Services: Conflicts Between Security Features and Publish/Subscribe Communication Protocols

Edward Komp, Victor Frost, and Martin
Kuehnhausen

ITTC-FY2010-TR-41420-19

February 2010

Project Sponsor:
Oak Ridge National Laboratory
Award Number 4000043403

Table of Contents

Table of Contents	i
List of Figures	i
Abstract	1
I. Introduction	1
II. Background	1
A. TSSN	1
1) Security Requirements	2
2) Asynchronous Communication Requirements	3
B. WSA-based Implementation	3
III. Implementation Conflicts.....	3
A. Server Operations Obscured	4
B. Reversal of Client and Server Roles.....	5
C. Multiple Subscribers, Different Security Policies.....	5
IV. Resolution	5
V. Conclusions.....	6
Acknowledgment	6
References.....	6

List of Figures

Figure 1: Top-level view of the TSSN architecture.....	2
--	---

Implementing Web Services: Conflicts Between Security Features and Publish/Subscribe Communication Protocols

Edward Komp, Victor Frost, *Fellow, IEEE*, and Martin Kuehnhausen, *Student Member IEEE*

Abstract - While on the surface the combination of software components that adhere to associated standards should lead to rapid and successful system implementation. However, issues can arise when integrating independently defined software subsystems. Here conflicts are discussed that arose when integrating elements from the Web Services Architecture[1] (WSA) led by the World Wide Web Consortium[2] (W3C), specifically publish/subscribe communication and service security. Unfortunately, the various standard components are seldom completely independent, and when separate components are jointly deployed unanticipated interactions sometimes cause significant problems at implementation time. The nature of the conflicts is discussed within the context of a specific system implementation the Transportation Security Sensor Network (TSSN) and interim solutions presented.

Index Terms— Eventing, Publish/Subscribe, Security, Service-Oriented-Architecture, SOA

I. INTRODUCTION

Divide and conquer is a standard engineering practice for large, complex systems. The system is often repeatedly subdivided into (nearly) independent components, allowing groups to work independently and in parallel on subtasks. This basic principle is fundamental to large ongoing specification efforts such as the Web Services Architecture[1] (WSA) led by the World Wide Web Consortium[2] (W3C). Unfortunately, the various components are seldom completely independent, and when separate components are jointly deployed unanticipated interactions sometimes cause significant problems at implementation time.

We have experienced an example of conflict between independently defined subsystems when jointly deployed, in our design and implementation of the Transportation Security Sensor Network[3] (TSSN). In this paper we provide a brief overview of our system, followed by a detailed description of the conflicts between security features and the publish/subscribe communication protocol, and finally techniques to resolve these conflicts.

II. BACKGROUND

A. TSSN

Monitoring cargo movements along trusted corridors

requires coordinated application of sensing, communications, as well as the integration of shipment and other associated cargo information. To realize a trusted corridor a Transportation Security Sensor Network (TSSN) has been designed, implemented and tested in the field [4] to provide the required visibility into cargo shipments.

The system is composed of three major geographically distributed components. The Mobile Rail Network (MRN) consists of container seals that communicate over a wireless network to a reader when an event occurs, e.g., seal open, a sensornet collector node that interfaces to the reader, processes events, determines which events need to be communicated to a virtual network operations center (VNOC), and the mode of communications (e.g., GSM or satellite). The second component is the VNOC, which accepts messages from the MRN, obtains associated cargo information from a remote trade data exchange and then combines the information (e.g., nature of the event, location of the event, and cargo manifest) into an alarm message that is sent (by e-mail or SMS) to appropriate decision makers. The third component is the trade data exchange that contains the shipment information and other associated cargo information. A goal of the effort was to create technologies that will allow continuous monitoring of containers by leveraging communications networks, sensors as well as trade and logistics data within an environment composed of multiple enterprises, owners, and operators of the infrastructure.

The resulting technologies must be open and easy to use, enabling small and medium sized enterprises (SMEs) to obtain the associated economic and security benefits. Thus a standards web-based open system is preferred. The architecture developed as part of this effort uses existing software components in addition to those specifically developed for the TSSN. The infrastructure is based on W3C and OASIS Web Service specifications; including service discovery, services are described using Web Service Description Language (WSDL), and Client/Server communication based on Simple Object Access Protocol (SOAP). Figure 1 provides a top-level view of the TSSN architecture.

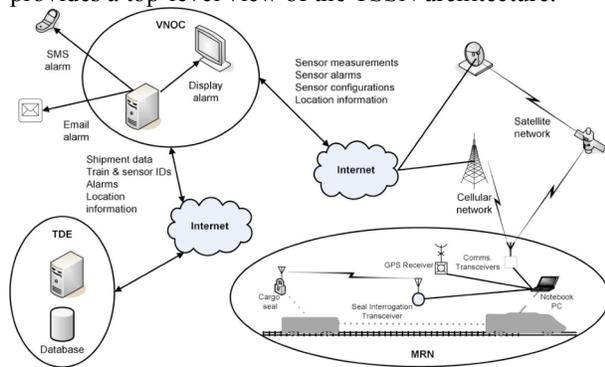


Figure 1

The TSSN has been tested in the field where the VNOc was located in Lawrence, Kansas, alarms generated on the train were sent to the MRN located in the locomotive, which forwarded relevant alarms to the VNOc. When the VNOc received an alarm, it contacted the TDE located at Overland Park, Kansas, to get the cargo information. The VNOc then combined the sensing and the shipment information to generate the alarm messages that were then communicated to a set of interested parties. See [4] for a discussion of TSSN field trials.

1) Security Requirements

Our application includes services and clients hosted by independent businesses often exchanging sensitive data. Therefore, robust and flexible security features were an integral portion of our design. It was clear that a single system wide security policy would be entirely inadequate to satisfy our goals.

As described in the following section, the Web Services Security[5] specifications provide support for

the range of capabilities required in our system definition.

The Web Services Security model includes many features that are particularly important to our application:

- Does not rely on a single, application-wide security policy. Each service is allowed to define a security policy to match the sensitivity of data exchanged and other system constraints, such as bandwidth limitations.
- Provides a mechanism for each service to publish its security policy in a formal standard syntax, so that clients do not require details of service implementation.
- Does not demand (though allows for) a single, centralized supplier of authorization.
- Permits each easy extension of both services and clients.

In our distributed application some services must operate within tight processor and/or bandwidth constraints. A general comprehensive security policy, predicated on relatively large bandwidths among members, probably would not fit within the operating constraints of some services in our system. However, even in constrained environments, it is not acceptable to ignore security issues. The structure defined in WS-Security (WSS) specification allows each service to define precisely the amount of security, and implementation guidelines, to a fine degree. The WSS specification supports the attachment of security constraints at a variety of levels within the service definition including, service-wide policy, specific policy per operation, and distinct policies for different connections to the same service. The specification further defines how overlapping policies are to be integrated. This flexibility in defining the security policy for a service allows one to precisely define the level of security for a wide variety of services and to address changing demands if additional capabilities (operations) are added to an existing service.

The security policy(ies) enforced by a service can be specified in the WSDL (Web Services Definition Language) [6] description that formally defines a web service interface. The security policy is a critical aspect of a service interface that must be understood by any potential client. In earlier system implementations

this information was generally expressed by any number of informal and ad hoc techniques, such as extra documentation and/or direct person-to-person communication between service provider and client. Web Services Policy 1.5 [7] defines a formal language to precisely define a service's security requirements, and to incorporate it into the web service interface definition. This is a major advancement for facilitating inter-operation of a service and its clients. This is particularly important when the service author and clients belong to independent business organizations.

2) *Asynchronous Communication Requirements*

A second fundamental aspect of our system is the ability to contact a diverse group of users when specific sensor events occur. For example, if a security lock is compromised on a container, we may need to immediately contact the carrier of the container, public first responders in the area in which the event occurs, as well as the owner of the specific container. Asynchronous notification does not directly fit the traditional client-server model. For many applications, some variation of client polling can be used in place of asynchronous notification. This approach is widely used in web applications, for example with RSS feeds. However, for our TSSN application, timely delivery of alerts and alarms is critical to utility of the system. Some portions of the TSSN network have very limited bandwidth and possibly long communication delays, further discouraging the usage of polling.

Fortunately, the Web Services Architecture includes the WS-Eventing [8] specification that defines a protocol for one Web service ("subscriber") to register interest in another Web service ("publisher"). In this document we will refer to this capability as the "publish/subscribe communication protocol". The publisher disseminates information to the subscriber(s) by sending one-way messages. The specification further defines mechanisms to dynamically add and remove members from the subscriber list; and support for complex publisher-subscriber topologies by allowing an "event source (to) delegate subscription management to another Web service"[8].

B. *WSA-based Implementation*

Adopting the Web Services Architecture as the basis for the TSSN dramatically reduced the effort we needed to both design and implement the infrastructure. Simply asserting that all services and clients would be WSA compliant eliminated the need to design and provide implementation for significant portions of our system. For example, our design simply stated that an alarm service would provide notification of the occurrence of an alarm condition via a publish/subscribe protocol (using WS-Eventing). We did not need to design the actual message protocols, and determine how they would be realized over various transport layers; nor address implementation issues of buffering, timeouts, etc.

Furthermore, a variety of open source and vendor-supplied implementations of various portions of the WSA are available. We chose to use the open source, Apache Axis2 [9], web services software stack in our implementation of TSSN. This middleware dramatically reduced the amount of code to be written by our team. In particular, the Axis2 program suite includes modules providing implementation for the two specifications, WS-Security and WS-Eventing, that are the subjects of this paper.

The Axis2 module, Rampart, is "a module based on Apache WSS4J to provide WS-Security features"[10]. This module places handlers in the pre-dispatch phase of Axis2 message handling, with independent configurations for each service on the server side. In addition, the Rampart module can automatically extract and enforce the security policy(ies) specified in the service definition. This feature ensures that the security requirements of a service always match the service's published security policy. In addition, the service provider is able to alter the security policy enforced for the service operations by simply modifying its service definition – without making source code changes.

The Axis module, Savan, is an "implementation of WS-Eventing specification ...designed as a general publisher/subscriber tool"[11]. When any service engages the Savan module, it provides implementation for a number of additional operations including Subscribe, Unsubscribe, Renew, and GetStatus.

III. IMPLEMENTATION CONFLICTS

In the design process, we did not specifically address

how security issues for the publish/subscribe communications would be handled. Security issues for servers and clients were considered independently in the context of the WS-Security specification and implementation. This separation is an example of the divide and conquer engineering practice described earlier. The decision to consider these aspects of the design independently was concretely supported by the web services architecture documentation. The WS-Eventing submission begins with the statement: “This specification specifically relies on other Web service specifications to provide secure, reliable, and/or transacted message delivery and to express Web service and client policy.” [12]

Preliminary implementation tests were encouraging. We created a simple client/server test case. First, the service interface was formally defined in WSDL 2.0 (Web Service Description Language)[6]. In this first version, no security policy was defined in the WSDL for the service; and programmers generated a successful implementation for the service and a client based on the formal interface definition. In the next step, a security policy was attached to some operations defined for the service. The service was rebuilt against the modified WSDL definition, and immediately enforced the stated security policy on the selected operations. The (unmodified) client was able to access only the operations for which no security policy was enforced. With minor modifications to the client, to provide the necessary security credentials demanded by the modified service, the client was able to invoke all service operations.

In summary, this test indicated that a service could be implemented without regard to the service policy demanded for its operations. In addition, different security policies could be assigned at a very fine grain down to different operations within the same service.

Unfortunately, when we proceeded to integrate these concepts into the implementation of TSSN we encountered serious unexpected difficulties. The TSSN includes services that are also clients to other services. For example, the VNOC alarm processor service sends email messages containing sensor alarm notification coupled with relevant cargo information. In order to do so, this service must solicit information from both the MRN and TDE services. So, the VNOC alarm

processor service also performs a client role to these two other services. Because alarms occur asynchronously, the MRN alarm processor uses the publish/subscribe protocol for exchanging information with its clients. It was very difficult to realize independent security policies for the VNOC alarm service and MRN alarm service for which it is a client.

This problem arises because of the interactions between the WS-Eventing specification (for the publish/subscribe communication) and WS-Security for the two services. For this discussion we label a service that is also a client to another service using the publish/subscribe protocol, as a *service with asynchronous client*.

A. Server Operations Obscured

When a service engages the Savan module to support WS-Eventing, Savan inserts handlers into the pre-dispatch phase of operations destined to the service, and implicitly provides features of the WS-Eventing specification for this service. The task of these pre-dispatch handlers is to detect and handle operations defined in the WS-Eventing specification. These operations include: Subscribe, Unsubscribe, Renew, GetStatus, however, these operations never appear in the WSDL for the user-defined service. So, these operations are “obscured”, that is, do not appear in the external, WSDL definition of the user-defined service. This has major impact on the specification and implementation of security for these operations as described in the following paragraph.

As demonstrated by our early test cases, using WSDL to formally define the service interface, including the security policy(ies) associated with its operation set, is a very powerful abstraction. Clearly, providing a standard, public definition of the interface promotes interoperability. In addition, middleware tools (we use, Apache Axis2) provide support for securing messages. If a service (client) engages the Axis2 Rampart module, handlers are inserted in the pre-dispatch phase to satisfy the security policy expressed in the service WSDL. The Rampart module automatically handles signing messages, encrypting/decrypting messages, applying and verifying timestamps, etc. This significantly reduced the coding effort for servers and clients. Perhaps, even more importantly, modification of security policy attached to service operations may require no modification of the service or client code,

since securing messages can be handled by the underlying Rampart module.

Since these eventing operations do not appear in the WSDL, there is no place to express the specific security policy for these operations. This has adverse effects for both the publisher (service) and subscriber (client).

Although the service author cannot attach security policy to specific WS-eventing operations in the WSDL, it is possible to attach a security policy at the service level. Specification at this level, attaches the same security policy to every operation of the service. Sometimes this is an acceptable compromise, but in general, one wants the finer level of security control that WS-policy provides for other services. In our application, we require higher levels of security for an entity requesting subscription to a sensor's alarm events, than to other more general status operations to determine if a sensor is active, etc.

For a service with asynchronous client assigning a service level security policy implicitly applies the same policy to both the service side *and* client side operations. This effectively requires this service to utilize the security policy of the service supplying it with asynchronous input for its own operation set. This is completely counter to the WS-Security intention to allow each service independently define its own security policy.

On the client (subscriber) side, obscuring these additional service operations also makes it difficult to ensure that the client satisfies the security policy enforced by the server. The service stub generator, used to generate a framework for a client for the service incorporates code to direct the Rampart module on the client-side to satisfy the security policy appearing the service WSDL for each visible operation. Since the eventing specific operations do not appear in the service WSDL, this behavior is not available.

On the client side, the explicit server eventing operations are further hidden from the client author. The client code must include an external library (provided with the Savan module) that provides an abstract functional interface providing the WS-Eventing capabilities. Internally the Savan module invokes the Subscribe (and related) operations provided by the service. Unfortunately, Savan completely ignores security policy required by the service it contacts.

B. Reversal of Client and Server Roles

For the delivery of asynchronous messages from server to client, Savan effectively reverses the roles of the participants. The server (publisher) delivers a notification to the client (subscriber) by invoking a client operation. The name of the client operation corresponds to the notification type. So, for notification, the server requests the named operation to be performed by the client.

From the perspective of security management, the client must enforce the appropriate security policy for the incoming operation message. The client now controls the level of security for the delivery of asynchronous messages. It is much more appropriate for the server, who is the owner of the information being distributed, to control the level of security used for this message transfer.

C. Multiple Subscribers, Different Security Policies

As described in the previous section, the Savan module permits the client (subscriber) to define the security policy for the delivery of asynchronous messages. In addition to mis-assigning security responsibility, this role reversal may make it (nearly) impossible for the server to successfully deliver notification to all clients. Multiple clients may subscribe to a server for the same notification (a core feature of WS-Eventing), and each could enforce a different security policy in its implementation of the associated notification operation. The Savan module provides no support for delivering notification to different subscribers with different security policies, so special care must be taken to ensure that every client specifies the same security policy for notification operations.

IV. RESOLUTION

As described in section III, use of the Savan module to implement WS-Eventing seriously restricts the independence of security policy for servers and clients.

Initial attempts to specify security policy for publishing services failed. The current release of Savan for subscription support, ignores security policy required by the service it contacts. Since these operations are invoked below the level of the user-written client code, there is no obvious way to ensure that the service provider's security requirements are

satisfied. In order to satisfy the security requirements for subscription in our system, we modified the Savan module, and added a new method to allow the client to transfer to Savan the service security policy.

If every subscriber to a server providing asynchronous notification is purely a client, then many of security policy conflicts can be resolved, though, informally, by the server describing the security policy it will apply to notification messages it sends; and expect all clients to respect this agreement. Unfortunately, this workaround significantly diminishes many of the advantages of WS-Security related to formally publishing security policy in the service description (in WSDL). In our experience, both server and client were required to embed in its implementation details of this security policy.

In the case of more complex clients, such as a server with asynchronous client, the challenges became more complex. For services of this type, it is necessary to distinguish those operations that define the interface of this service to its clients, from the operations added to handle the notifications sent to it. We devoted large amounts of time to testing and debugging to eventually generate services meeting specific security constraints for our application. The goal of generating a solution with a flexible and extensible set of security policies was compromised at the critical juncture where multiple services communicate asynchronously.

V. CONCLUSIONS

We were able to successfully field the TSSN as a set of web services and ensured that all links satisfied our security constraints. But to do so, we had to require that otherwise independent services shared the same security policy for their respective operations; and we had to embed security policy decisions directly into the implementation of some services. These compromises restrict the flexibility and extensibility of TSSN.

To fully resolve these issues, we believe that aspects of the WS-Eventing specification, and the Savan module implementation, in particular, need to be reviewed from the perspective of interactions with WS-Security.

The specifications should more clearly identify who is responsible for the definition of security policy for the delivery of messages from publisher to subscriber. The publisher of the information invokes an operation of the

subscribing (client) service, thereby deferring the security policy to the operation owner, or client. However, since the publisher is the actual owner of the information being disseminated, it seems that it should maintain responsibility for the service policy to enforce when delivering messages.

In addition, the implementers of the Savan module should review the implementation strategy for the WS-Eventing specifications to ensure that it respects the security policies expressed in the WSDL definition of any service that activates the Savan module. In addition, consideration should be given to make all WS-Eventing operations, such as subscribe, and renew, explicit in the WSDL interface for each service activating the Savan module. This is important to allow the publishing service advertise (and enforce) specific security policy for these operations.

ACKNOWLEDGMENT

This work was supported in part by Oak Ridge National Laboratory (ORNL) Award Number 4000043403. This material is also partially based upon work supported while V. S. Frost was serving at the National Science Foundation.

REFERENCES

- [1] David Booth, et al. "Web Services Architecture, W3C Working Group Note." <http://www.w3.org/TR/ws-arch/>, W3C 2004, October 2009.
- [2] W3C. <http://www.w3.org/>, W3C 2009, October 2009.
- [3] Martin Kuehnhausen, "Service Oriented Architecture for Monitoring Cargo in Motion Along Trusted Corridors," Master's thesis, University of Kansas, July 2009.
- [4] D. T. Fokum, V. S. Frost, D. DePardo, M. Kuehnhausen, A. N. Ogunu, L. S. Searl, E. Komp, M. Zeets, D. D. Deavours, J. B. Evans, and G. J. Minden, "Experiences from a Transportation Security Sensor Network Field Trial." to appear Proc. 3rd IEEE Workshop on Enabling the Future Service-Oriented Internet-Towards Socially-Aware Networks (EFSOI 09), Honolulu, Hawaii, USA, Nov. 2009.
- [5] Anthony Nadalin (ed.) et al. "Web Services Security: SOAP Message Security 1.1." <http://docs.oasis-open.org/wss/v1.1/>, OASIS Open, Feb. 2006.
- [6] Roberto Chinnic (ed.) et al. "Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language," <http://www.w3.org/TR/wsd120/>, W3C 2007, October 2009.
- [7] Asir S Vadamuthu (ed.) et al. "Web Services Policy 1.5 – Framework." <http://www.w3.org/TR/2007/REC-ws-policy-20070904/>, W3C September 2007, October 2009.
- [8] Don Box (ed.), et al. "Web Services Eventing (WS-Eventing)," <http://www.w3.org/Submission/WS-Eventing/>, W3C March 2006, October 2009.

- [9] "Apache Axis2/Java - Next Generation Web Services," <http://ws.apache.org/axis2/>, Apache Software Foundation 2009, October 2009.
- [10] "Securing SOAP Messages with Rampart", <http://ws.apache.org/axis2/modules/rampart/1.0/security-module.html>, May 04, 2007.
- [11] "Apache Savan/Java", <http://wso2.org/projects/savan/java>, WSO2 Inc., 2009.
- [12] Don Box (ed.), et al. "Web Services Eventing (WS-Eventing): Introduction." <http://www.w3.org/Submission/WS-Eventing/>, W3C March 2006, October 2009.

The University of Kansas



Technical Report

Framework for Analyzing SOAP Messages in Web Service Environments

Martin Kuehnhausen and Victor S. Frost

ITTC-FY2010-TR-41420-20

March 2010

Project Sponsor:
Oak Ridge National Laboratory
Award Number 4000043403

Copyright © 2010:
The University of Kansas
2335 Irving Hill Road, Lawrence, KS 66045-7559
All rights reserved.

Table of Contents

Table of Contents.....	i
List of Figures.....	i
List of Tables.....	i
Abstract.....	1
I. Introduction.....	1
II. Problem Area.....	1
A. Logging.....	1
B. Analysis.....	1
C. Visualization.....	2
III. Related Work.....	2
A. Logging Systems.....	2
B. Web Service Specifications.....	2
IV. Proposed Solution.....	4
A. Logging.....	4
B. Log Parser.....	5
C. Visualization.....	7
V. Results.....	7
A. Short Haul Rail Trial.....	7
VI. Conclusion.....	9
Acknowledgment.....	9
References.....	9

List of Figures

Figure 1: Geographically distributed services.....	2
Figure 2: Framework overview: logging, log parser and visualization (from left to right).....	4
Figure 3: SOAP message (left) to Log parser classes (right) comparison.....	4
Figure 4: Two transmit-receive pairs (red and green).....	5
Figure 5: A message couple (red).....	5
Figure 6: Log file and service interaction visualization showing individual services grouped by their physical location.....	6
Figure 7: Complete message flow from the MRN Sensor Node to the VNOC Alarm Reporting Service.....	6
Figure 7(a): Initial alarm notification within the MRN from the Sensor Node service to the Alarm Processor service.....	6
Figure 7(b): Alarm notification from the MRN Alarm Processor to the VNOC Alarm Processor.....	6
Figure 7(c): Shipment information and route query from the VNOC Alarm.....	6
Figure 7(d): Alarm message with additional shipment information forwarded to the VNOC Alarm Reporting service.....	6
Figure 8: Request performance from [16].....	7
Figure 9: Network transmission and processing comparison from [16].....	8
Figure 9(a): Network transmission performance from [16].....	8
Figure 9(b): Processing performance from [16].....	8
Figure 10: System alarm notification performance from [16].....	8

List of Tables

Table I: XPath Expressions for WS-Addressing.....	5
---	---

Framework for Analyzing SOAP Messages in Web Service Environments

Martin Kuehnhausen *Graduate Student Member, IEEE* and Victor S. Frost, *Fellow, IEEE*

Abstract—Service Oriented Architectures can be quite complex which makes managing and monitoring them hard. Message exchanges as well as complete message flows are important when it comes to analyzing the performance of systems and identifying problems. The use of SOAP as the common message format for web services enables interoperability and extensibility. Furthermore it can be used for logging and analysis because its components follow web service specifications. We present a framework that allows for logging, analyzing and visualizing these messages across a distributed system.

The proposed framework consists of a flexible logging module that captures incoming and outgoing messages of a web service. The log parsing library provides various methods for normalizing, correlating across geographically distributed sites and analyzing messages. The visualization tool is able to display relationships between the web services as well as message flows.

In utilizing this adaptable framework for analyzing SOAP messages it is possible to overcome the challenges of complexity and disparity that monitoring and management approaches face in web service environments

Index Terms—Logging, Log analysis systems, Visualization, Transport protocols, Data communication, Software engineering

I. INTRODUCTION

THE use of Service Oriented Architectures (SOA) in application systems is widespread. The idea is to implement specific functionality in web services that communicate with each other using standardized interfaces. Message exchanges in general use the flexible SOAP message format [1]. This has a number of advantages as for instance message routing and security are available as extensions to it. However, in many cases and especially in geographically distributed systems as shown in Figure 1, there exists no simple way to analyze and visualize the message flow through the entire system as well as measuring performance of components in order to identify potential bottlenecks.

Manuscript received March 2, 2010.

M. Kuehnhausen and V. S. Frost are with the Information and Telecommunication Technology Center, The University of Kansas, Lawrence, KS, 66045, USA; e-mail: mkuehnha@ittc.ku.edu and frost@ittc.ku.edu

This work was supported in part by Oak Ridge National Laboratory (ORNL)—Award Number 4000043403. This material is also partially based upon work supported while V. S. Frost was serving at the National Science Foundation.

This paper describes a framework that is able to analyze a variety of timing measurements such as message transmission and service processing times. Furthermore it correlates messages which can then be used to analyze message flows. Finally a graphical user interface is presented that allows the visualization of individual system components and their message interactions.

II. PROBLEM AREA

Web service environments consist of disparate systems and technologies. In order to efficiently monitor and analyze message flows the following areas need to be addressed.

A. Logging

Chuvakin et al. [2] explain that logs often contain “valuable information about systems, networks, and applications”. In particular logging is important when it comes to auditing because laws often mandate so-called audit trails, for example in the health industry and the banking sector. Another important observation that Chuvakin et al. make is that logging in distributed systems needs to be addressed differently as web services “are by their nature distributed across multiple systems, disparate technologies and policies, and even organizational domains”. This means that logging needs to be addressed on a global basis either with a centralized logging server or with a common logging format that is used by each individual service.

B. Analysis

Apart from auditing purposes log file analysis is often used to study or detect failures in systems. Furthermore usage and bandwidth monitoring in distributed systems is important for load balancing in order to keep costs down. Lim et al. [3] outline particular aspects that need to be considered in logging systems, in their case an enterprise telephony system, that allow efficient and effective analysis.

They also note that one of the problems is that data in logs is often unstructured and it is therefore hard to automatically discover patterns. However, this is often solved by log preprocessing such as message normalization and clustering. After cleaning the logs data mining approaches such as finding frequent item sets, frequency analysis and anomaly detection may be applied. Furthermore data in logs can be correlated and dependencies of messages can lead to message flow analysis. In terms of performance analysis measurements such as processing and transmission times are important.

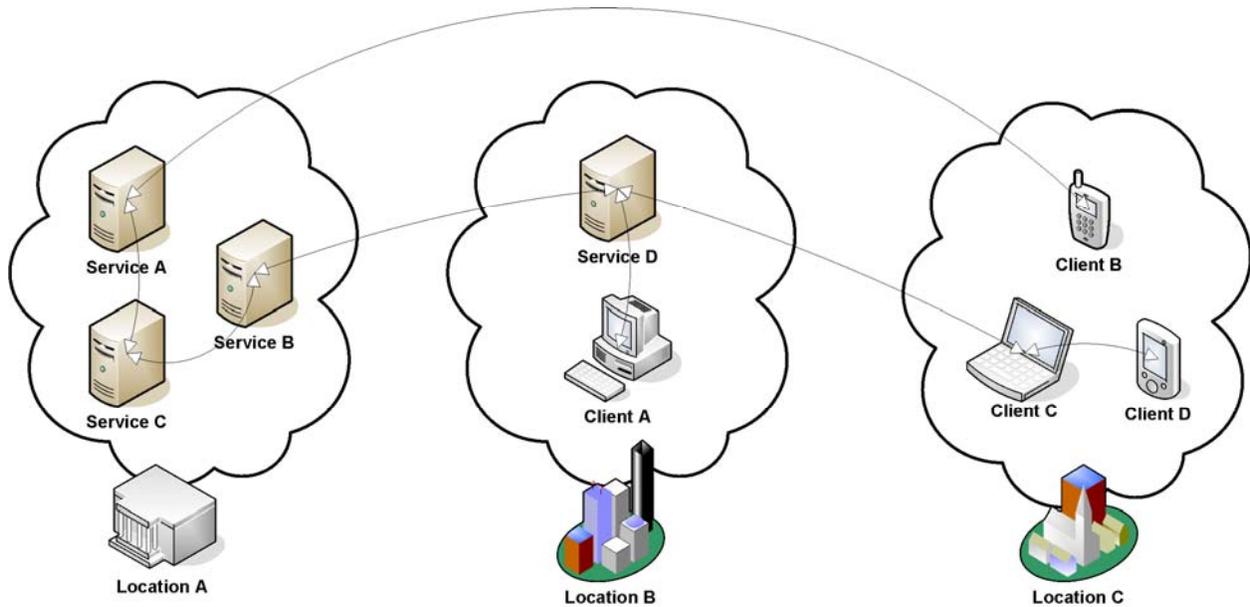


Fig. 1. Geographically distributed services

C. Visualization

Since there is often an abundance of data present it is not necessarily easy to grasp important aspects or quickly analyze data. This is especially true regarding message flows. While time measurements and message statistics can easily be represented using a table format, sequences and dependencies need more complex structures such as graphs and trees.

III. RELATED WORK

A. Logging systems

Cinque et al. [4] propose a system that is based on specific logging rules that allow “effective dependability evaluation of complex systems”. By defining a common set of rules such as service start, service end, entity (resource) interaction start and end an observer is able to follow certain event flows. Given estimated computation duration and considering potential timeouts the system generates alerts whenever it detects a problem in an event flow.

Vaarandi [5] introduces two compact log file analysis tools called Simple Logfile Clustering Tool (SLCT) and LogHound which apply two of the most common approaches to log analysis. SLCT uses a density based clustering method that is able to detect outliers while LogHound “employs a frequent itemset mining algorithm”.

Makanju et al. [6] provide an overview of “network information visualization tools” as well as propose their own version called LogView. They are using a variety of techniques such as plots and treemaps to visualize network traffic, intrusion detection and application logs.

Logging web sessions that track specific users generates a lot of data. In contrast to other logging systems the focus here lies on analyzing flows and user behaviors. Session Viewer [7] by Lam et al. is a visual tool that consists of various panels for data aggregation, cluster and flows analysis. This allows a statistical overview as well as detailed analysis of so-called

session logs.

An approach that applies directly to web services is proposed by Simmonds et al. [8]. They describe a runtime monitoring system based on a subset of UML 2.0 Sequence Diagrams which are used for checking conditions and messages exchanges. After the constraints are correctly defined in sequence diagrams they are handed over to a “monitoring manager” that receives events from a “message manager” and checks those against the events and flows specified. Simmonds et al. also discuss various related work in terms of the difference in “offline” and “online” monitoring as well as “global” and “local” property checking. Their proposed system is able to efficiently check the correctness of events and flows at runtime.

Service Oriented Architectures are inherently based on web service specifications. These specifications clearly define standard properties and functionality of web services that implement a particular specification. Within a logging system this leads to the ability of extracting information according to these standards independent of how they are implemented and in a flexible and extensible manner. This is not necessarily true in most logging system that depend on a particular log format or are specifically built for one particular purpose. Whenever more information needs to be captured the log format needs to be adapted in these systems while using a web service based approach this is not necessary.

B. Web Service Specifications

In web service environments there are two common types of message exchanges. The first is a one-way message transfer from a service to another service or client which is often used in notification scenarios. The second is a common two-way exchange in which a request message is sent to a particular service, the service then fulfills the request and sends out a response accordingly. Apart from these basic message exchange patterns it is possible to set up a subscription system

and then have a web service automatically deliver messages to a client.

It is important to note that the messages themselves can be encrypted in order to protect private information which can pose issues when trying to analyze message flows. In the following the web service specifications related to message exchanges, subscriptions and security are discussed.

1) *WS-Addressing*:

The *WS-Addressing* core specification [9] and its SOAP binding [10] defines how message propagation can be achieved using the SOAP message format. Usually the transport of messages is handled by the underlying transport protocol but there are several advantages of storing this transport information as part of the header in the actual SOAP message. For example, it allows the routing of messages across different protocols and management of individual flows and processes within web services.

WS-Addressing uses so-called *EndPointReferences* which are a collection of a specific address, reference parameters and associated metadata that further describe its policies and capabilities. The *Addressing Header* header fields defined by the specification are the following:

- *To* which represents the destination of the message
- *From* contains the source, a so-called *EndPointReference*
- *ReplyTo* specifies that in case of a response, a message is supposed to be sent to this *EndPointReference*, which might be different from the *From* field
- *FaultTo* defines the *EndPointReference* for the fault message in the case of an error
- *Action* identifies the purpose of the message, in particular the web service operation, and is the only required field
- *MessageID* uniquely identifies every message
- *RelatesTo* references the *MessageID* of the request message in request-response message exchanges; the relationship can also be specified explicitly by defining a so-called *RelationShipType*

2) *WS-Eventing*:

In order to allow for subscriptions to web services, the *WS-Eventing* specification [11] has been defined. It describes the process of establishing subscriptions as well as how the subsequent publications are delivered to the subscribers. The specification relies on *WS-Addressing* for the routing of messages. The two main components of a subscription in this specification are the *Subscribe* and the *SubscribeResponse* message. After subscriptions have been created, publications will be sent out accordingly.

a) *Subscribe*

The client that wants to subscribe to a particular web service needs to define the following:

- The *Action* field of the *WS-Addressing* header is set to <http://schemas.xmlsoap.org/ws/2004/08/eventing/Subscribe>

rite

- *ReplyTo* is the *EndPointReference* that receives the response to this subscription request
- A *MessageID* that uniquely distinguishes multiple requests from the same source
- *EndTo* defines an *EndPointReference* that is used when the subscription ends unexpectedly
- *Delivery* contains the *EndPointReferences* that are to receive the publications
- An *Expires* field that defines the expiration time of the subscription
- *Filter* that by default defines an *XPath* expression as the *Dialect*, but could be any form of expression that is applied to potential publications in order to filter them

b) *SubscribeResponse*

The response to a subscription request is generated by the so-called *subscription manager*. It sends back a message with these fields:

- The *Action* field of the *WS-Addressing* header is set to <http://schemas.xmlsoap.org/ws/2004/08/eventing/SubscribeResponse>
- *RelatesTo* specifies the subscription request that this is a response to
- *SubscriptionManager* that contains its own *Address* and the unique Identifier for the subscription
- An *Expires* field that defines the expiration time of the subscription

The *WS-Eventing* specification also offers message constructs for the renewal, status retrieval and unsubscribing of subscriptions. Additionally a so-called *subscription end* message is automatically generated by the service that publishes information in order to notify subscribers of errors or other reasons for it being unable to continue the subscription.

It has to be noted that without additional specifications like *WS-ReliableMessaging* the delivery of publications is based purely on best effort and is not guaranteed.

3) *WS-Security*

The *WS-Security* specification [12] deals with the many features needed to achieve so-called *end-to-end* message security. This provides security throughout message routing and overcomes the limitations of so-called *point-to-point transport layer security* such as HTTPS. Furthermore, the specification aims to provide support for a variety security token formats, trust domains, signature formats and encryption technologies.

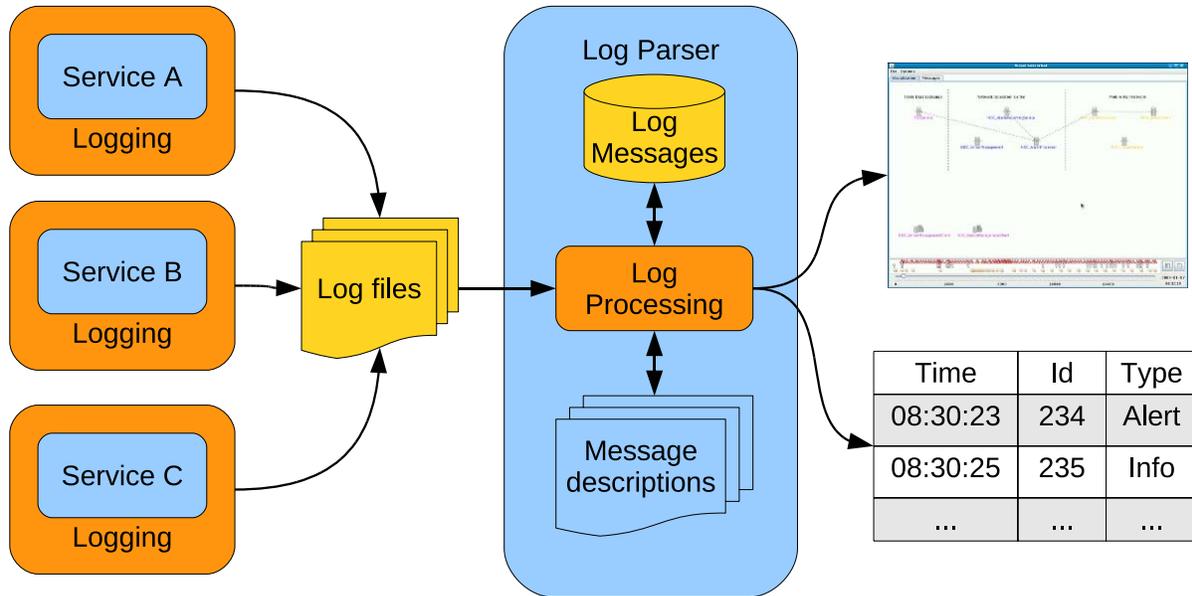


Fig. 2. Framework overview: logging, log parser and visualization (from left to right)

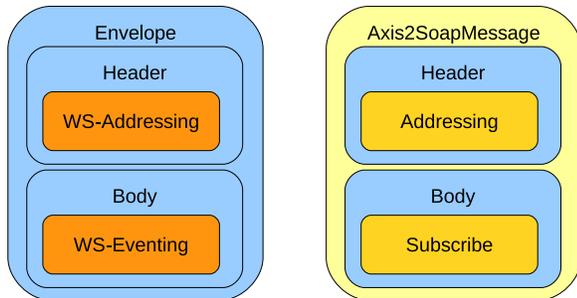


Fig. 3. SOAP message (left) to Log parser classes (right) comparison

IV. PROPOSED SOLUTION

The framework developed here allows the capture and analysis of SOAP messages in one and two-way communications as well as subscriptions. Logging these messages that are based on web service specifications and extracting information from them overcomes common problems previous approaches faced such as the necessity of a common log format across different platforms and the inability to extend the system later to log more information without breaking existing functionality.

The individual components are shown in Figure 2 and are explained in the following sections. Note that the logging part is positioned in between the message sender/receiver and the security layer which makes it possible to capture message information without compromising security functionality in the web service environment.

A. Logging

The logging component that was developed as part of this framework was initially designed to support a particular

system, the Transportation Security SensorNet (TSSN) [13]. However, it can easily be adapted to other systems. The TSSN is based on the Axis2 web service stack which uses modules for implementing web service specifications. While modules for WS-Addressing and WS-Eventing come as part of Axis2, a module that implements the desired logging functionality had to be developed separately.

1) Logging Module

The logging module as described in the following provides extensive logging capabilities to the *web services*. It was *engaged* during development and testing of the entire TSSN system since it logs all messages that are sent and received. In addition, it also writes the raw contents of the SOAP messages that are sent and received into log files. In particular the following information is captured:

- *Time* when the message was sent or received
- *Service* which is used
- *Operation* that is being executed
- *Direction* of the message, which can be either incoming or outgoing. Note that there are special directions that
- deal with incoming and outgoing faults.
- *From* address of the message
- *Reply to* address that may differ from the *From* address
- *To* address of the message
- *Schema* element that is being “transported” as part of the operation containing the request parameters or the response elements
- *Size* of the message in bytes
- *Message* which represents the entire SOAP message in a readable form

TABLE I
XPATH EXPRESSIONS FOR WS-ADDRESSING

XPath expression	Method equivalent
//To/text()	getTo()
//ReplyTo//Address/text()	getReplyTo()
//From/Address/text()	getFrom()
//MessageID/text()	getMessageId()
//RelatesTo/text()	getRelatesTo()
//Action/text()	getAction()

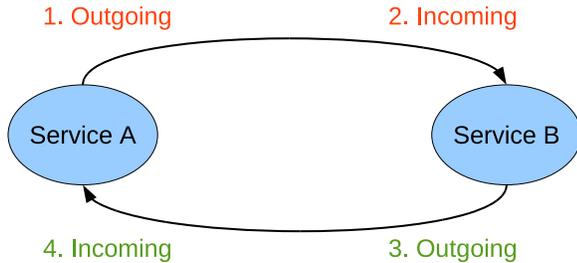


Fig. 4. Two transmit-receive pairs (red and green)

In terms of analyzing the TSSN and its performance the logging module was engaged in all services. More information on the findings can be found in V.

2) Addressing:

An implementation of the *WS-Addressing* specification as described in III-B1 comes as part of the *addressing* module in the Axis2 core. It fully supports all components of the standard and its *ReplyTo* and *RelatesTo* fields are used among other things to allow for asynchronous communication in the TSSN.

3) Savan:

The *Savan* module enables web services and clients in Axis2 to make use of various forms of subscription mechanisms as defined by the *WS-Eventing* specification (see III-B2).

B. Log Parser

The log parser enables parsing, processing and merging of log files. It transforms the raw SOAP messages into Java elements that can then be filtered and analyzed.

1) Abstraction Layer Model:

Since SOAP is essentially XML, information from the so-called log messages can be retrieved using *XPath* [14] *path expressions*. For this purpose the *log parser* provides an object *abstraction layer model* that corresponds to the specific parts in the SOAP message.

An example mapping is shown in Figure 3. It displays the structure of the original SOAP message (for more information on the individual SOAP messages see III-B) on the left and the equivalent *log parser* objects on the right. Note that the corresponding objects highlighted in yellow are actual classes while the *Header* and *Body* are not abstracted separately.

The log parser objects would then provide access to their properties using *XPath expressions*. In this case they correspond to their respective *web service* specifications but they could also be defined according to the XML schema definitions of any other element. For example, for the *WS-*

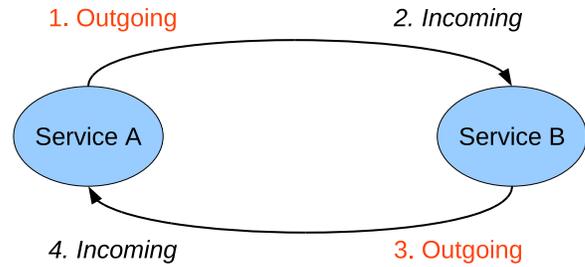


Fig. 5. A message couple (red)

Addressing (see III-B1) equivalent object the path expressions in Table I are used:

This mapping process is easily defined because it corresponds to the web service specifications and allows for an in-depth analysis of the messages that are sent and received.

2) Message Types:

Since the *logging* module is enabled on both ends of a message exchange, the *log parser* is able to correlate messages. In order to do this it makes use of the so-called *message id* that is provided by the *WS-Addressing* specification. It has to be noted that a requirement for the following analysis is that the times on both ends of the message transfer are synchronized. Within the TSSN system this is done using NTP [15] but it is also possible with GPS.

Without this assumption the computed times are questionable and represent an estimation at best. The following two types of message associations are present in the log files:

a) Transmit-Receive Pair:

Whenever a message is sent out by a particular client or service it is captured by the logging module. The receiving service logs the message as well but as an incoming message. The content of the message is essentially the same which can also be seen by the fact that they have the same message id. The outgoing and the incoming message are combined and form what is called a transmit-receive pair.

This allows us to compute the message transfer or so-called transmit time which describes how long it takes to transmit the message from one entity to another using the following equations:

$$transmitTime_1 = time_{2.Incoming} - time_{1.Outgoing} \quad (1)$$

$$transmitTime_2 = time_{4.Incoming} - time_{3.Outgoing} \quad (2)$$

As shown in Figure 4 the *log parser* automatically detects the *transmit-receive* pairs and stores them in a particular list for further analysis.

b) Message Couple:

The most common message exchange pattern is the *In-Out* pattern. It defines *request-response* based message transfers which the *log parser* calls *message couples*. A single message

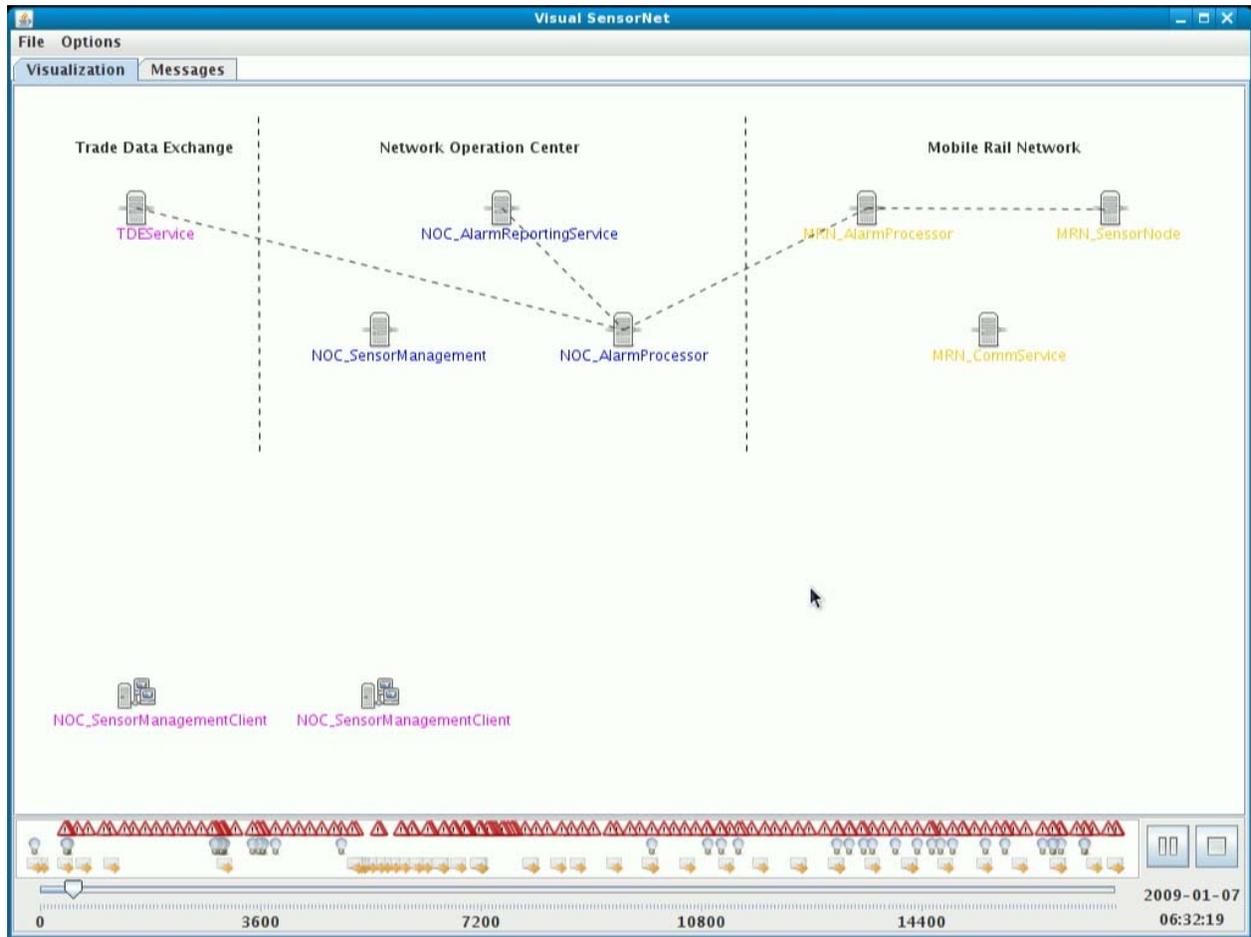


Fig. 6. Log file and service interaction visualization showing individual services grouped by their physical location

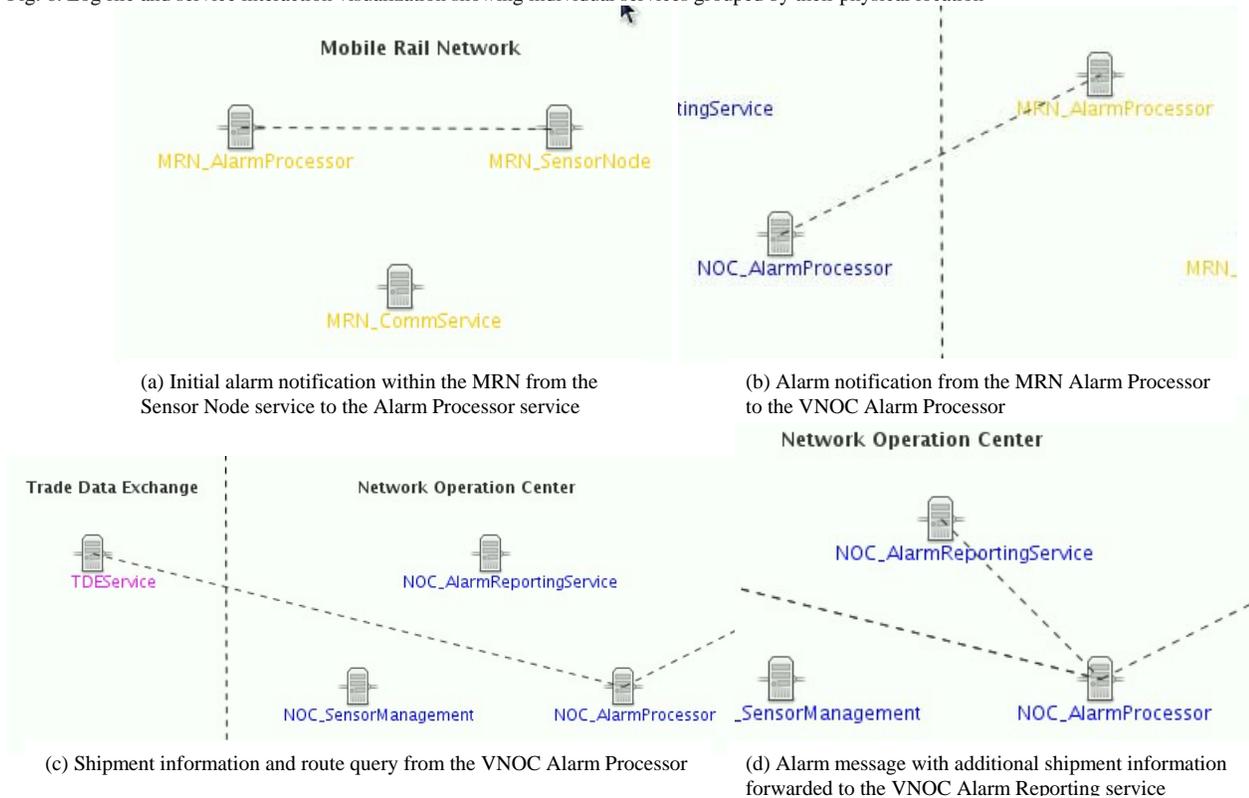


Fig. 7. Complete message flow from the MRN Sensor Node to the VNOc Alarm Reporting Service

couple consists of two messages, the outgoing request and the outgoing response on the receiving entity, which is shown in Figure 5. They can be correlated using the *WS-Addressing* specification. The request will carry a *message id* and the response a so-called *relatesTo id* in addition to its own unique *message id*.

Note that a *message couple* can also be seen as a combination of two *transmit-receive* pairs. This relationship is extremely useful in computing measures such as round trip and processing times:

$$\text{roundTripTime} = \text{time}_{4,\text{Incoming}} - \text{time}_{1,\text{Outgoing}} \quad (3)$$

$$\text{processingTime} = \text{time}_{3,\text{Outgoing}} - \text{time}_{2,\text{Incoming}} \quad (4)$$

The *log parser* provides functionality to associate messages and analyze complete end-to-end message flows. More details on the performance measurements and test results can be found in V.

C. Visualization

In order to understand the message flows better without needing too much of a technical background, a visualization tool was developed. It makes use of the *log parser* to display services, clients and messages that are present in log files.

The user is able to load and merge log files to create a visualization of services and clients as shown in Figure 6. Lines connecting individual services appear as communication occurs. The layout of these services is defined according to their membership in a particular *service cloud*. Furthermore, any point in time that is part of the log files can be “jumped to” using the time line. It displays significant events in the log files:

- *Alarms, alerts* and *sensor node events* with a warning sign
- *Requests* such as location retrieval with a light bulb sign
- *Control messages* such as *start monitoring* with a message sign

The scenario that was captured by the log files can also be played back in portions or in its entirety. Using the visualization tool, it is therefore possible to analyze service interactions and message flows conveniently. An example message flow is shown in Figure 7.

V. RESULTS

The framework described here enabled analysis of field trials of the TSSN. The Transportation Security SensorNet [13] uses a Service Oriented Architecture approach for monitoring cargo in motion along trusted corridors. The complete system provides a web services based sensor management and event notification infrastructure that is built using open standards and specifications. Particular functionality within the system has been implemented in web services that provide interfaces according to their respective web service specifications. This web services based implementation allows for platform and programming language independence and offers compatibility and

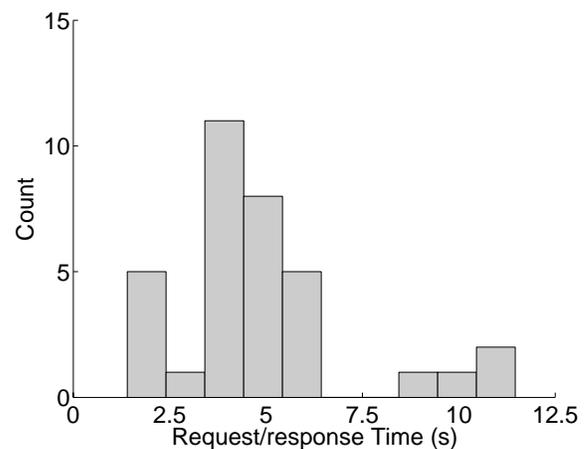


Fig. 8. Request performance from [16]

interoperability with other systems.

Furthermore, unlike existing proprietary implementations the TSSN allows sensor networks to be utilized in a standardized and open way through web services. Sensor networks and their particular communication models led to the implementation of asynchronous message transports in SOA and are supported by the TSSN.

An in-depth analysis of the real world scenarios that were performed to test the TSSN is given by [16]. For the tests the *Trade Data Exchange* was deployed in Overland Park, the *Virtual Network Operation Center* at the *University of Kansas* in Lawrence and the *Mobile Rail Network* either on a truck or on a train. Note that in both cases the communication between the *Mobile Rail Network* and the *Virtual Operation Center* was established using a GSM modem. The main findings are as follows:

A. Short Haul Rail Trial

This more advanced scenario was performed by deploying the *Mobile Rail Network* on a locomotive of a train along with sensors attached to containers for it to monitor. The train traveled approximately 35 kilometers during the trip, from a *rail intermodal facility* to a rail yard.

The system faced some of the same issues as during the truck trials such as loss of GPS, GSM and sensor communication. The data that was collected however shows that again the *Transportation Security SensorNet* was able to deal with them and send out alarm notifications reliably. The log files were analyzed using the *log parser* and led to the following:

1) Message Counts:

During the *short haul rail trial* the *Sensor Node* reported 546 alerts¹ to the *Alarm Processor*. After filtering 131 alarms were sent to the *Alarm Processor* at the *Virtual Network Operation Center*. For 63 of them, shipment information was queried from the *Trade Data Exchange* and 33 were stored as so-called *validated alarms*. All of the 131 alarms that the *Alarm Processor* received were sent out to *Alarm Reporting service* which notified the according contacts via SMS and

¹ Alerts were generated manually by opening and closing an electronic seal in the locomotive and automatically by things such as losing a GPS fix

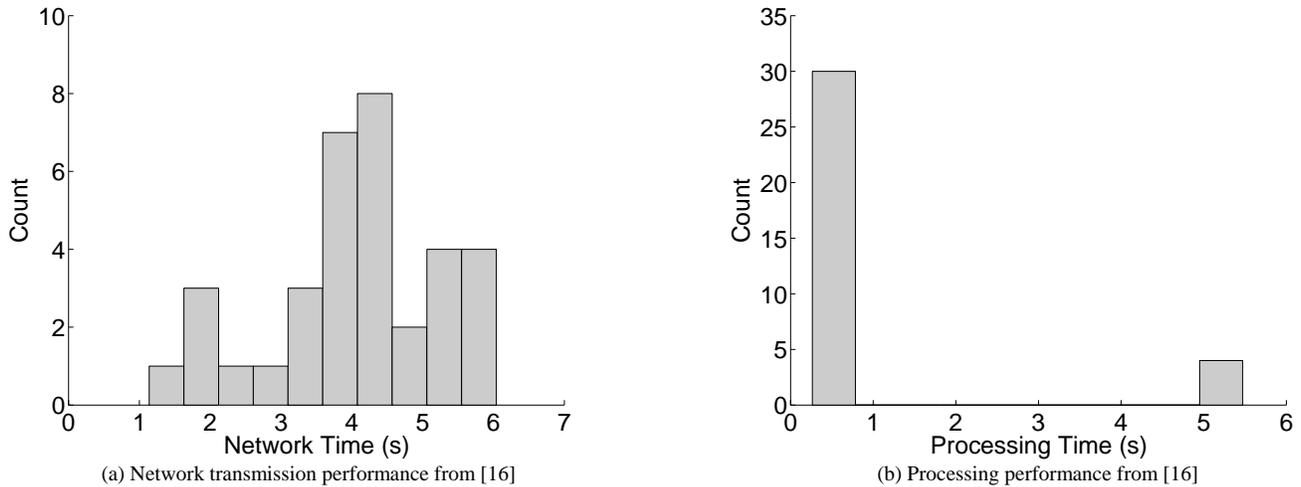


Fig. 9. Network transmission and processing comparison from [16]

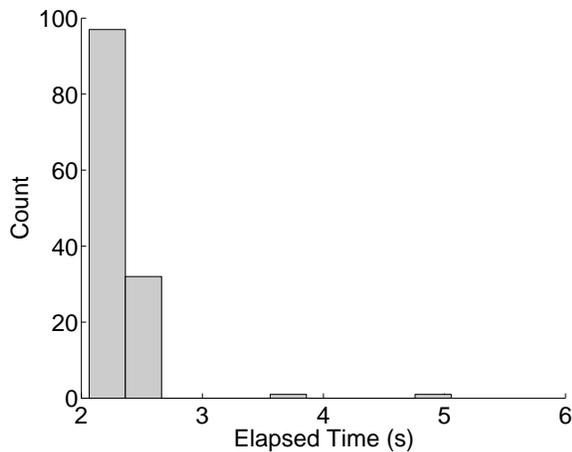


Fig. 10. System alarm notification performance from [16]

email. There were also 30 inquiries from the TDE for the location of the *Mobile Rail Network*.

2) Message Sizes:

Looking at the communication between the *Virtual Network Operation Center* and the *Mobile Rail Network* one can notice the following pattern. So-called *control messages* such as *startMonitoring* or *getLocation* are always initiated at the *Virtual Network Operation Center*. Since these messages usually transmit only a small functional request, the average message size is around 690 bytes. On the other hand, Alarms are always sent from the *Mobile Rail Network* and contain a lot of valuable information. Hence the average message size is about 1420 bytes.

a) Request Performance:

As shown in Figure 8, the time it took for messages from the *Virtual Network Operation Center (Sensor Management)* to send requests to the *Mobile Rail Network* (either *Sensor Node* or *Alarm Processor*) and receive a response was about 4.4 seconds on average. The fastest request was answered in 0.9 seconds while the slowest took about 11 seconds.

Overall these numbers meet the expectations of the transportation industry. Performing a location inquiry given an

average train speed of 30 km/h and 60 seconds to retrieve the location, the actual position and the reported one may differ by as much as 500 meters. However, the *Transportation Security SensorNet* provides location information in less than 5 seconds resulting in a maximum difference of just 41.7 meters.

The bottleneck here is the message transmit time as defined in Equation 1. As shown in Figure 9, processing on the *Sensor Node* took only 0.6 seconds on average whereas about 85% of the time is spent on message transmission. This percentage is likely to increase when switching to satellite communication instead of communicating with the GSM modem which was used in the trials.

b) Alarm Notification Performance:

Because of the problems with the clock drift, the measured times for messages coming from the *Mobile Rail Network* going to the *Virtual Network Operation Center* are unreliable. However, taking our previous findings about the request performance the time for this particular transmission can be estimated using the average round trip and the processing times:

$$\bar{t} = \frac{1}{n} \sum_{i=1}^n (\text{roundTripTime}_i - \text{processingTime}_i) \quad (5)$$

$$= \frac{4.4\text{seconds} - 0.6\text{seconds}}{2} \quad (6)$$

$$= 1.9\text{seconds} \quad (7)$$

Given this estimate transmit time t , we can compute the total time it takes from for an alarm to go through the entire TSSN as shown in 10.

This includes the times from the *Sensor Node* to the *Alarm Processor* at the *Mobile Rail Network*, the approximated transmit time of 1.9 seconds, and the time from the *Alarm Processor* to the *Alarm Reporting service* at the *Virtual Network Operation Center*. On average this yields about 2.1seconds with the fastest time being just over 1.9 seconds

and the slowest around 4.9 seconds.

Both, the road test with trucks and the short haul rail trial can be called successful because they displayed the capabilities of the TSSN, its good performance and that the functionality implemented in the web services worked. In particular, two of its main capabilities, location inquiry and alarm notification were extensively demonstrated. Furthermore, the time it took from registering alerts, propagating them through the *Transportation Security SensorNet* and sending out notifications accordingly is under 5 seconds and significantly smaller than expected for such a complex system.

The framework was used to capture and analyze the results presented here. For example we were able to break down the measurements into processing and transmission times as well as determine the performance of individual web services and the total alarm notification performance. Furthermore being able to visualize message flows helped identify problems and locate the particular part that was causing it.

VI. CONCLUSION

Capturing SOAP messages directly and using them as the basis for log analysis has the advantage of being a more structured approach because the SOAP messages adhere to specific web service specifications. This allows convenient mappings from the SOAP messages to elements defined in the specification and vice versa.

In this paper we presented a flexible framework for analyzing SOAP Messages in web service environments. The proposed solution consists of three parts. First, a logging module that can be attached to web services in order to capture SOAP messages and log their send and receive times. Second, a log parsing and processing library that allows for efficient correlation of messages, message flows and analysis. Finally, a visualization tool that provides convenient visual analysis of service interactions capabilities.

ACKNOWLEDGMENT

This work was supported in part by Oak Ridge National Laboratory (ORNL) Award Number 4000043403. This material is also partially based upon work supported while V. S. Frost was serving at the National Science Foundation.

REFERENCES

- [1] Y. Lafon and N. Mitra, "SOAP version 1.2 part 0: Primer (second edition)," W3C, W3C Recommendation, Apr. 2007, <http://www.w3.org/TR/2007/REC-soap12-part0-20070427/>.
- [2] A. Chuvakin and G. Peterson, "Logging in the age of web services," *IEEE Security and Privacy*, vol. 7, pp. 82–85, 2009.
- [3] C. Lim, N. Singh, and S. Yajnik, "A log mining approach to failure analysis of enterprise telephony systems," in *Dependable Systems and Networks With FTCS and DCC, 2008. DSN 2008. IEEE International Conference on*, June 2008, pp. 398–403.
- [4] M. Cinque, D. Cotroneo, and A. Pecchia, "A logging approach for effective dependability evaluation of complex systems," in *Dependability, 2009. DEPEND '09. Second International Conference on*, June 2009, pp. 105–110.
- [5] R. Vaarandi, "Mining event logs with slct and loghound," in *Network Operations and Management Symposium, 2008. NOMS 2008. IEEE*, April 2008, pp. 1071–1074.
- [6] A. Makanju, S. Brooks, A. Zincir-Heywood, and E. Miliotis, "Logview: Visualizing event log clusters," in *Privacy, Security and Trust, 2008. PST '08. Sixth Annual Conference on*, Oct. 2008, pp. 99–108.
- [7] H. Lam, D. Russell, D. Tang, and T. Munzner, "Session viewer: Visual exploratory analysis of web session logs," in *Visual Analytics Science and Technology, 2007. VAST 2007. IEEE Symposium on*, 30 2007–Nov. 1 2007, pp. 147–154.
- [8] J. Simmonds, Y. Gan, M. Chechik, S. Nejati, B. O'Farrell, E. Litani, and J. Waterhouse, "Runtime monitoring of web service conversations," *IEEE Transactions on Services Computing*, vol. 99, no. PrePrints, pp. 223–244, 2009.
- [9] M. Gudgin, M. Hadley, and T. Rogers, "Web services addressing 1.0 - core," W3C, W3C Recommendation, May 2006, <http://www.w3.org/TR/2006/REC-ws-addr-core-20060509>.
- [10] M. Gudgin, M. Gudgin, M. Hadley, T. Rogers, T. Rogers, and M. Hadley, "Web services addressing 1.0 - SOAP binding," W3C, W3C Recommendation, May 2006, <http://www.w3.org/TR/2006/REC-ws-addr-soap-20060509>.
- [11] D. Box, L. F. Cabrera, C. Critchley, F. Curbera, D. Ferguson, S. Graham, D. Hull, G. Kakivaya, A. Lewis, B. Lovering, P. Niblett, D. Orchard, S. Samdarshi, J. Schlimmer, I. Sedukhin, J. Shewchuk, S. Weerawarana, and D. Wortendyke, "Web services eventing (ws-eventing)," W3C, W3C Member Submission, Mar. 2006, <http://www.w3.org/Submission/2006/SUBM-WS-Eventing-20060315/>.
- [12] K. Lawrence, C. Kaler, A. Nadalin, R. Monzillo, and P. Hallam-Baker, "Web Services Security: SOAP Message Security 1.1 (WS-Security 2004)," OASIS, OASIS Standard, Feb. 2006, <http://docs.oasis-open.org/wss/v1.1/wss-v1.1-spec-os-SOAPMessageSecurity.pdf>.
- [13] M. Kuehnhausen, "Service Oriented Architecture for Monitoring Cargo in Motion Along Trusted Corridors," Master's thesis, University of Kansas, Jul. 2009.
- [14] S. Boag, A. Berglund, D. Chamberlin, J. Sim'eon, M. Kay, J. Robie, and M. F. Fernández, "XML path language (XPath) 2.0," W3C, W3C Recommendation, Jan. 2007, <http://www.w3.org/TR/2007/REC-xpath20-20070123/>.
- [15] D. Mills, "Network Time Protocol (NTP)," RFC 958, Internet Engineering Task Force, September 1985, obsoleted by RFCs 1059, 1119, 1305. [Online]. Available: <http://www.ietf.org/rfc/rfc958.txt>
- [16] D. T. Fokum, V. S. Frost, D. DePardo, M. Kuehnhausen, A. N. Oguna, L. S. Searl, E. Komp, M. Zeets, J. B. Evans, and G. J. Minden, "Experiences from a Transportation Security Sensor Network Field Trial," Information Telecommunication and Technology Center, University of Kansas, Lawrence, KS, Tech. Rep. ITTC-FY2009-TR-41420-11, June 2009.

The University of Kansas



Technical Report

An Open System Transportation Security Sensor Network: Field Trial Experiences

Daniel T. Fokum, Victor S. Frost, Daniel DePardo,
Martin Kuehnhausen, Angela N. Oguna,
Leon S. Searl, Edward Komp, Matthew Zeets,
Daniel D. Deavours, Joseph B. Evans,
and Gary J. Minden

ITTC-FY2010-TR-41420-21

March 2010

Project Sponsor:
Oak Ridge National Laboratory
Award Number 4000043403

Copyright © 2010:
The University of Kansas
2335 Irving Hill Road, Lawrence, KS 66045-7559
All rights reserved.

Abstract

Cargo shipments are subject to hijack, theft, or tampering. Furthermore, cargo shipments are at risk of being used to transport contraband, potentially resulting in fines to shippers. The Transportation Security Sensor Network (TSSN), which is based on open software systems and Service Oriented Architecture (SOA) principles, has been developed to mitigate these risks. Using commercial off-the-shelf (COTS) hardware, the TSSN is able to detect events and report those relevant to appropriate decision makers. However, field testing is required to validate the system architecture as well as to determine if the system can provide timely event notification. Field experiments were conducted to assess the TSSN's suitability for monitoring rail-borne cargo. Log files were collected from these experiments and postprocessed. We present the TSSN architecture and results of field experiments, including the time taken to report events using the TSSN as well as on the interaction between various components of the TSSN. These results show that the TSSN architecture can be used to monitor rail-borne cargo.

CONTENTS

I	Introduction	1
II	System Architecture	2
II-A	Trade Data Exchange	4
II-B	Virtual Network Operations Center	4
II-C	Mobile Rail Network	5
II-C1	Mobile Rail Network Hardware	5
II-C2	Mobile Rail Network Software	7
III	Experiments	8
III-A	Road Test with Trucks	9
III-B	Short-haul Rail Trial	9
IV	Postprocessing of Experimental Data	13
V	Results	15
V-A	Road Test: Message Counts	16
V-B	Short-haul Trial: Message Counts	16
V-C	Network Time from VNOC to MRN to VNOC	17
V-D	Elapsed Time from Alert Generation to AlarmReporting Service	17
V-E	End-to-end Time from Event Occurrence to Decision Maker Notification	18
V-F	Modeling of Decision Maker Notification Time	19
V-G	Timing Analysis of Other TSSN Interactions	20
V-H	Message Sizes	21
VI	Refinements Based on Experimental Results	22
VII	Related Work	22
VIII	Conclusion	23
	Acknowledgments	23
	References	24

LIST OF FIGURES

1	Transportation Security Sensor Network (TSSN) Architecture	3
2	Virtual Network Operations Center Architecture	5
3	TSSN Collector Node Hardware Configuration	6
4	Container Seal	7
5	Mobile Rail Network Collector Node Architecture	8
6	Map of Road Test with Event Annotations	10
7	Logical Short-haul Rail Trial Configuration	10
8	Collector Node and Sensor Deployment During Short-haul Rail Trial	11
9	E-mail Message Received During Short-haul Trial	12
10	SMS Message Received During Short-haul Trial	12
11	LogParser Framework Showing Message Couples and Transmit-receive Pairs	14
12	Network Times from VNOG → MRN → VNOG	17
13	Sequence Diagram with Messages Involved in Decision Maker Notification	18

LIST OF TABLES

I	Summary of Time Statistics for Decision Maker Notification	18
II	Estimated Gamma Distribution Parameters for Time Taken Between Seal Events and Decision Maker Notification	20
III	Summary of Time Statistics for other TSSN Interactions	21
IV	Summary of Message Size Statistics	21

I. INTRODUCTION

IN 2006 the FBI estimated that cargo theft cost the US economy between 15 and 30 billion dollars per year [1]. Cargo theft affects originators, shippers, and receivers as follows: originators and receivers need a reliable supply chain to deliver goods in a timely and cost-effective manner. Shippers hold liability and insurance costs for shipments, which are proportional to the rate of theft. Finally, receivers are impacted by out-of-stock and scheduling issues due to cargo theft. Most non-bulk cargo travels in shipping containers. Container transport is characterized by complex interactions between shipping companies, industries, and liability regimes [2]. Deficiencies in the container transport chain expose the system to attacks such as the commandeering of a legitimate trading identity to ship an illegitimate or dangerous consignment, hijack, or the theft of goods. Insufficiencies in these areas can be overcome by creating secure trade lanes, or trusted corridors, especially at intermodal points, for example, at rail/truck transitions. Research and development is underway to realize the vision of trusted corridors.

The work described here focuses on: advanced communications, networking, and information technology applied to creating trusted corridors. The objective of the research is to provide the basis needed to improve the efficiency and security of trade lanes by combining real-time tracking and associated sensor information with shipment information. One crucial research question that must be answered in order to attain this objective is how to create open technologies that will allow continuous monitoring of containers by integrating communications networks, sensors as well as trade and logistics data. This integration must occur within an environment composed of multiple enterprises, owners, and infrastructure operators.

To achieve improved efficiency and security of trade lanes, we have developed a Transportation Security Sensor Network (TSSN) architecture, which uses Service Oriented Architecture (SOA) [3] principles, for monitoring the integrity of rail-borne cargo shipments. The TSSN is an open system where different components can be provided by different vendors. The TSSN is composed of a Trade Data Exchange (TDE) [4], Virtual Network Operations Center (VNOC), and Mobile Rail Network (MRN). The functions of each of these components are discussed in greater detail in Section II. The TSSN detects events, integrates the event type from the train in the field with logistics information, and then reports those events that are important to decision makers using networks with commercial links. Ideally, decision makers would be notified of events within 15 minutes [5] so that they can take effective action. For the TSSN to be deployed, we need to validate its architecture and understand the timeliness of the system response. Work by Arsanjani *et al.* [6] and Saiedian and Mulkey [7] shows that service oriented architectures introduce overhead. As a result, we want to determine whether an SOA-based system such

as the TSSN provides timely event notification. TSSN event notification is also impacted by unpredictable packet latency on commercial networks and the use of e-mail and/or Short Message Service (SMS) [8] for event notification. Thus, we have designed and implemented hardware and software needed to realize a prototype of the TSSN and carried out experiments [9] to characterize the system, particularly the end-to-end time between event occurrence and decision maker notification using SMS or e-mail as well as the impact of SOA overhead.

In this paper we provide a high-level description of the TSSN architecture and document two field experiments that were carried out to demonstrate that sensors and software based on an open service-oriented architecture can be used to monitor cargo in motion. Our experiments focussed on determining the time from event occurrence to decision maker notification as well as testing functionality between the component services of the prototype TSSN. Our experimental results show that decision makers can be notified of events on the train in a timely manner using the prototype TSSN architecture. The rest of this paper is laid out as follows: In Section II we describe the TSSN architecture including the components and the prototype hardware implementation. Section III discusses experiments conducted to assess the suitability of the TSSN system for cargo monitoring. In Section IV we discuss the framework used to postprocess the log files from our experiments. Section V presents empirical results showing the interaction between various components of the TSSN. Refinements to the TSSN given our field trial experiences are discussed in Section VI. In Section VII we present related research on monitoring trains and securing shipping containers in motion. Section VIII provides concluding remarks.

II. SYSTEM ARCHITECTURE

To achieve the vision of a trusted corridor we have designed and implemented a prototype of the Transportation Security Sensor Network (TSSN) architecture. The detailed architecture of the TSSN, including system extensibility, is found in [10], whereas this section gives an overview of the TSSN. The architectural details discussed here are important in understanding the experiments and results presented in Sections III and V, respectively.

The SOA and web services used in the TSSN enable the integration of different systems from multiple participating partners. Moreover, the use of SOA and web services enable data to be entered once and used many times. Using commercial off-the-shelf (COTS) hardware and networks, as well as an open systems approach, the TSSN is able to detect events and report those relevant to shippers and other decision makers as alarms. Furthermore, the TSSN supports multiple methods for notifying decision makers of system events.

communications are done using networks with commercial wireless link components. The TDE, VNOC, and MRN are examined in greater detail in the following subsections.

A. Trade Data Exchange

The Trade Data Exchange (TDE) contains shipping data and it interconnects commercial, regulatory and security stakeholders. The TDE is based on a “technology-neutral, standards-based, service-oriented architecture” [4]. The TDE is hosted on a server with a wired connection to the Internet. The TDE is geographically separated from the VNOC and responds to queries from the VNOC. The TDE also stores event messages sent by the VNOC. Finally, the TDE sends commands to start and stop monitoring at the MRN as well as to get the train’s current location.

In addition to the functions mentioned above, the TDE monitors the progress of shipment and other logistics information. The TDE captures commercial and clearance data including: the shipping list, bill of lading, commercial invoice, Certificate of Origin (for example, NAFTA Letter), and shipper’s export declaration. It also validates and verifies data to ensure accuracy, consistency, and completeness. The TDE monitors the progress of the documentation and notifies responsible parties when errors or incompleteness pose the threat of delaying a shipment. The TDE forwards notification to the customs broker to request verification of the trade origination documents. The customs broker accesses the TDE via the same portal to review and verify the trade documentation. Finally, the TDE allows for collaboration between participating shippers, third-party logistics providers, carriers and customs brokers to define and document business requirements and risk assessment requirements. Real-time cargo sensing capability is provided to the TDE via the TSSN. Data from the TDE is combined with event data from the MRN to provide the decision maker complete information concerning the alarm, e.g., cargo information, location, and nature of the event.

B. Virtual Network Operations Center

The Virtual Network Operations Center is the management facility of the TSSN [10] and it is also the shipper’s interface to the TDE. The VNOC can be the central decision and connection point for multiple MRNs. The VNOC consists of services that receive and process alarms from the MRN as well as services that notify decision makers of events. Fig. 2 summarizes the VNOC and its components.

The functions of the VNOC include: forwarding commands from a client to the MRN to start and stop sensor monitoring as well as to get the MRN’s current location, receiving *MRN_Alarms* from the MRN, obtaining event-associated cargo information from the Trade Data Exchange (TDE) in real time,

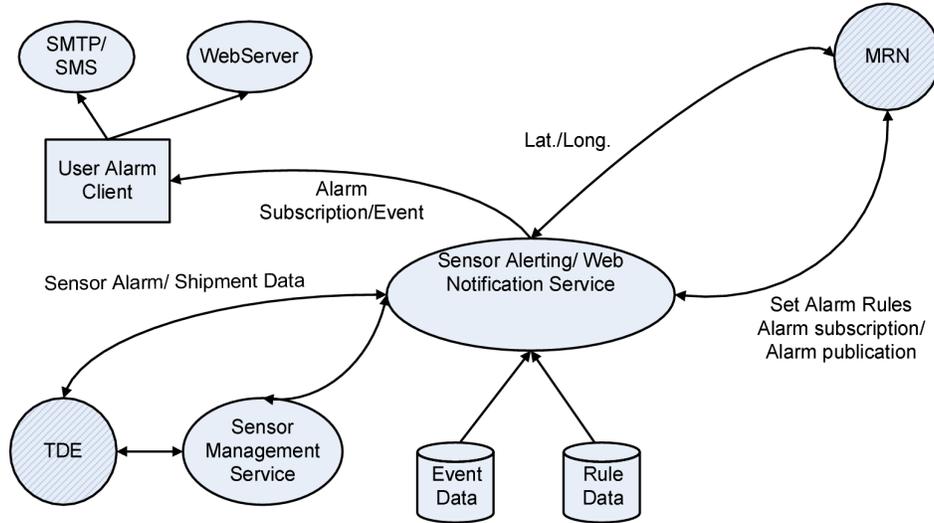


Fig. 2. Virtual Network Operations Center Architecture

and combining cargo information obtained from the TDE with an *MRN_Alarm* to form a *VNOC_Alarm* message that is sent (by SMS and/or e-mail to decision makers as shown in Figs. 9 and 10). A key role of the VNOC is getting the the right alarm information to the right personnel in a timely manner and also to prevent personnel from being overwhelmed by event messages. An AlarmProcessor service in the VNOC makes decisions, using rules, on which *MRN_Alarms* are forwarded to decision makers. For example, a low battery alarm is sent to technical staff, while an unexpected open/close event is sent to system security personnel. These decisions are made using a complex event processor, Esper [21], which takes into account shipping information as well as data (for example, geographical location) from current and past *MRN_Alarms*.

C. Mobile Rail Network

The MRN subsystem is located on the train and it consists of hardware and software. The prototype hardware and software architecture is described below.

1) *Mobile Rail Network Hardware*: The MRN subsystem hardware consists of a set of wireless shipping container security seals and a TSSN collector node. The collector node is composed of two major sections: an electronics suite mounted in the locomotive cab and a remote antenna assembly that is magnetically attached to the exterior of the locomotive. Fig. 3 summarizes the key components of the TSSN collector node.

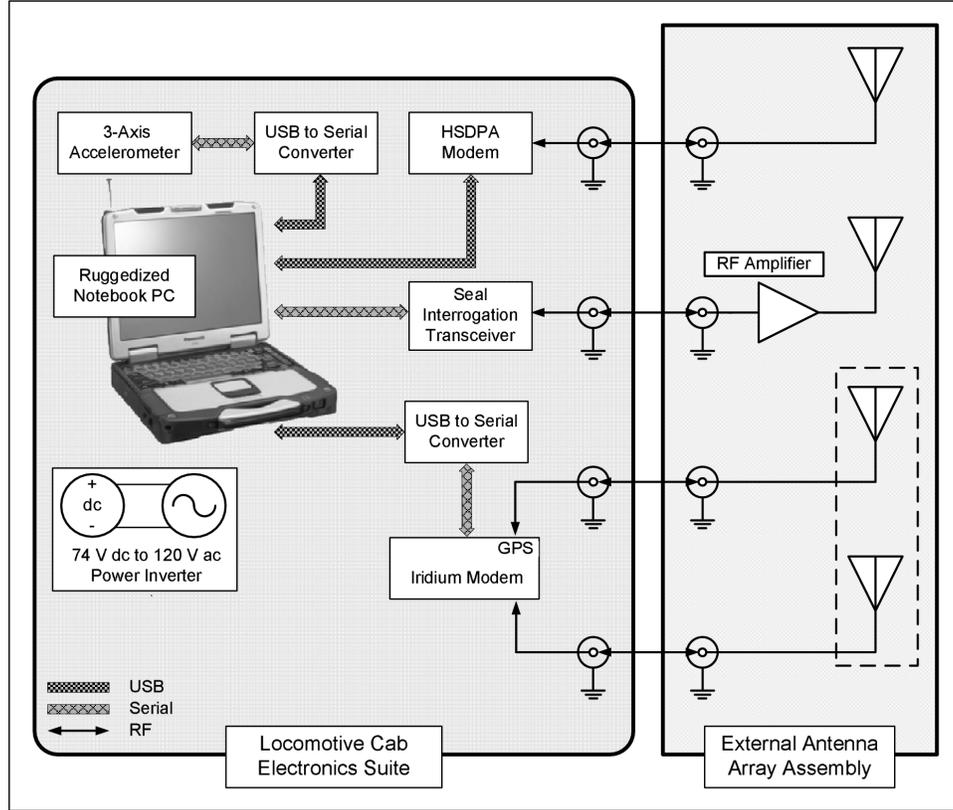


Fig. 3. TSSN Collector Node Hardware Configuration

The electronics suite contains a power inverter, a security seal interrogation transceiver, a computing platform, wireless data modems, a three-axis accelerometer, and a GPS receiver. The antenna assembly consists of three communications antennas, a GPS receiver antenna, and a bidirectional RF amplifier. Coaxial cables connect electronics suite devices to corresponding antennas.

Container physical security is monitored using a system that was originally designed for tanker truck security [19]. Container security is monitored with active and battery-powered container seals (sensors) equipped with flexible wire lanyards that are threaded through container keeper bar lock hasps as shown in Fig. 4. These seals had no support for multihop communications. The TSSN is designed to monitor and report security seal events including seal opened, seal closed, tampered seal, seal armed, seal missing, and low battery warnings. The seal interrogation transceiver (SIT) communicates with the container seals over a wireless network while the interrogation transceiver communicates with a notebook computer via a serial data connection. In order to conserve energy the container seals are asleep most of the time [22]. About every three seconds the seals listen for commands from the interrogation transceiver; however, the



Fig. 4. Container Seal

frequency at which the seals listen for commands is configurable. If the sensors are instructed to listen for commands more frequently then their battery lifetimes are reduced, whereas longer intervals between interrogations result in longer battery lifetimes [22].

Communication between the MRN and the VNOc is accomplished using a HSDPA cellular data modem. An Iridium satellite modem is also available and is intended for use in remote locations that lack cellular network coverage. The Iridium modem is a combination unit that includes a GPS receiver, which is used to provide the MRN with position information.

2) *Mobile Rail Network Software*: The prototype MRN software was implemented using a service-oriented architecture approach. The software consists of a SensorNode service, an AlarmProcessor service, and a Communications service. The SensorNode service finds and monitors sensors that have been assigned to its control. The SensorNode service manages several sensor software plug-ins, for example, a seal interrogation transceiver plug-in and a GPS device plug-in, that do all the work on behalf of the SensorNode service. During typical operation each container seal listens for interrogation command signals at regular intervals from the interrogation transceiver. In the event of a seal being opened, closed, or tampered with, the seal immediately transmits a message to the SensorNode service running on the Collector Node. The message contains the seal event, a unique seal ID, and event time. The SensorNode service passes the seal message as an *Alert* message to the service that has subscribed for this information.

The AlarmProcessor service determines messages from the SensorNode service that require transmission to the VNOc. Alarm messages include the seal event, event time, seal ID, and train's GPS location. The Communications service uses either HSDPA or Iridium for reporting events via the Internet to the

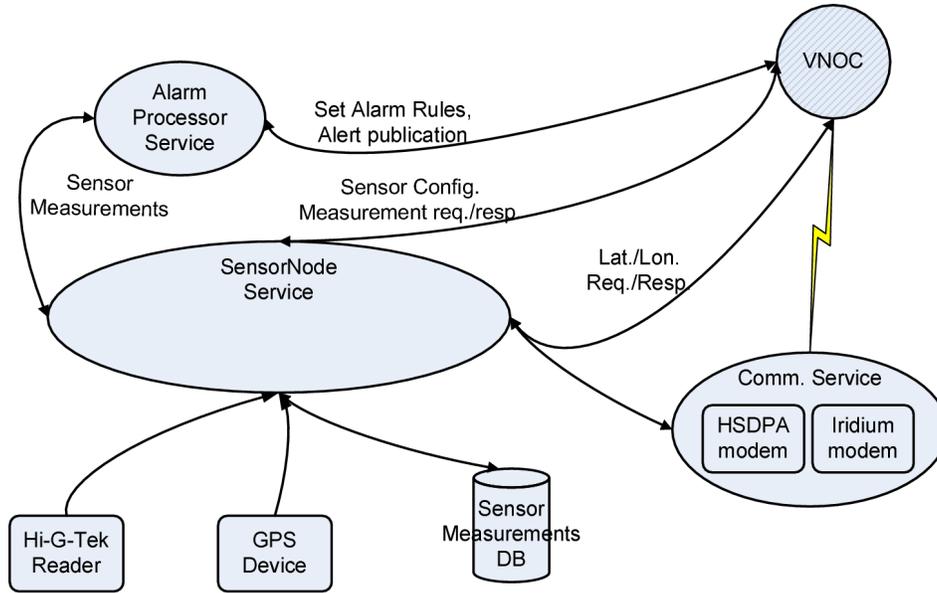


Fig. 5. Mobile Rail Network Collector Node Architecture

VNOC. Fig. 5 shows the key software functions of the MRN.

III. EXPERIMENTS

This section presents two experiments—a road test and the short-haul rail trial—conducted to assess the suitability of the TSSN architecture for cargo monitoring as well as to collect data that would be used to guide the design of future cargo monitoring systems. It is non-trivial to carry out experiments on moving freight trains; furthermore, as part of this effort we were limited to one chance to carry out experiments from a train. As a result, the TSSN architecture was tested in several static and some mobile tests, including the road test with trucks and the short-haul rail trial. In this section we present the experimental objectives, configuration, data collected during the tests, and issues that were encountered during the tests. The overarching goals of these experiments were to:

- Demonstrate the concept of using sensors, communications, and a service oriented architecture to monitor cargo in motion using the TSSN architecture.
- Determine the time from event occurrence to decision maker notification in a real field experiment.
- Verify proper operation of the prototype TSSN in a field environment. Proper operation means all messages were transmitted, received, and processed as expected and decision makers received all correct notification.

Thus, the following items were within scope of our experiments: the stability and timely performance of the communications protocols between TSSN component services, whereas the following items were out of scope: overall system robustness, whole train monitoring, energy consumption of the sensors, comprehensive security¹ issues, such as message spoofing, and decision maker response time given that event notification had been delivered.

A. Road Test with Trucks

The first experiment was conducted with two pickup trucks on local roads to validate the system operation and to determine if correct information is reported by the TSSN collector node, including valid GPS coordinates. One of the pickup trucks used in the test had the locomotive cab electronics suite in the truck bed, while both trucks had seals in their truck cabins so that seal open and close events could be emulated and reported. The VNOC was located in Lawrence, Kansas while the TDE was located in Overland Park, Kansas. The trucks were driven for approximately 1.5 hours over a 90 km route that began and ended in Lawrence, Kansas. The experiment route covered suburban and rural roads as well as state highways. During the experiment the seals were opened and closed at selected intersections along the test route that were easily identifiable on Google Maps [23].

Fig. 6 shows a trace of our route and the events overlaid on a Google map. The diamonds indicate an open event, tear drops a close event, circles with slashes across indicate a GPS lost signal, tacks indicate where the GPS signal was regained, a triangle with an exclamation sign indicates where HSDPA connectivity was lost, and the arrow indicates where HSDPA connectivity was regained.

B. Short-haul Rail Trial

Another experiment was carried out using a freight train travelling from an intermodal facility to a rail yard. Our objectives in this experiment were the following:

- To determine the performance of the prototype TSSN architecture when detecting events on intermodal containers in a real rail environment.
- To investigate if decision makers could be informed of events in a timely manner using SMS messages and e-mails.
- To collect data that will be used in a model to investigate system trade-offs and the design of communications systems and networks for monitoring rail-borne cargo.
- Evaluate the overall system performance to guide the future development of the TSSN architecture.



Fig. 6. Map of Road Test with Event Annotations

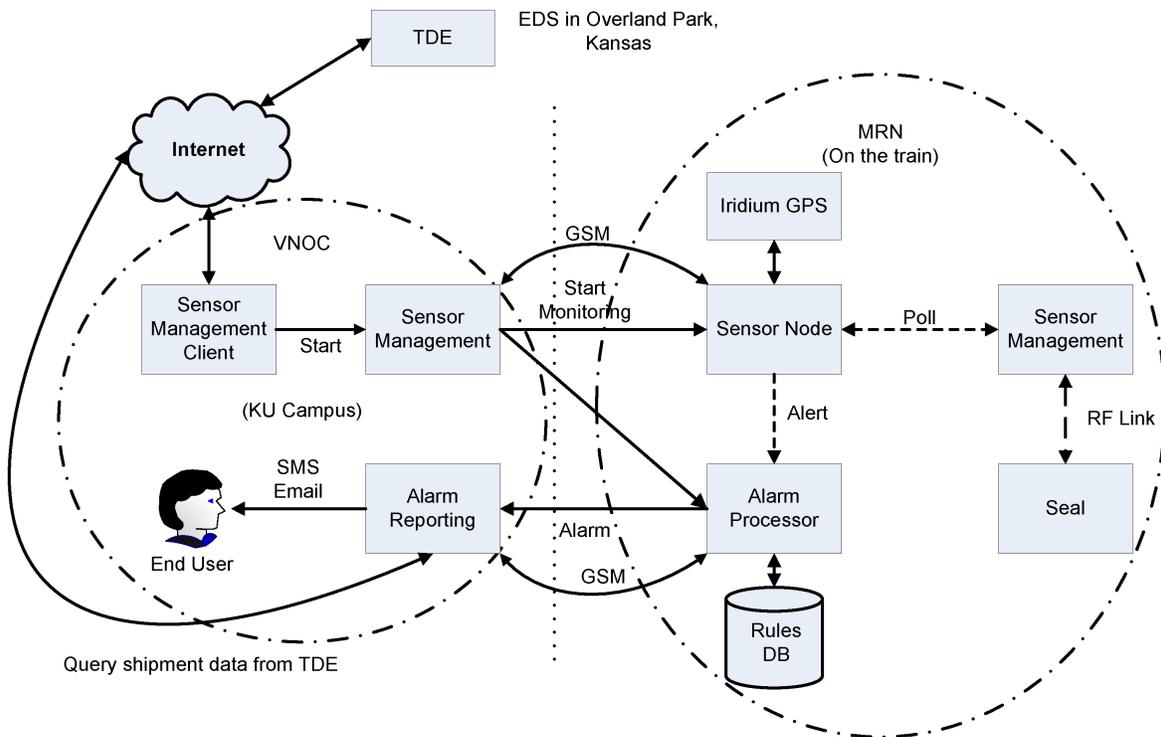


Fig. 7. Logical Short-haul Rail Trial Configuration

Fig. 7 shows the logical system configuration used in the short-haul rail trial. In this experiment the VNOc was located in Lawrence, Kansas, the TDE was located in Overland Park, Kansas, and the MRN was placed on the train. Within the MRN, the TSSN collector node was placed in a locomotive and

¹Comprehensive security issues are being addressed in the next version of the prototype.



Fig. 8. Collector Node and Sensor Deployment During Short-haul Rail Trial

used to monitor five seals. All communications between the MRN and the VNOC were passed through a Virtual Private Network (VPN) for message security. Prior to the start of the experiment prototype logistics data was added to the TDE to facilitate testing.

During the short-haul trial the train traveled for approximately five hours over a 35 km (22 miles) route. The route, which traversed both rural and urban areas, was relatively flat with a total elevation change of about 100 m. Fig. 8 shows a picture of the train used in the short-haul rail trial along with the arrangement of the sensors (wire seals). As shown in Fig. 8, the short-haul trial train was composed of well-cars with a mixture of empty cars, cars with a single container, or cars with double-stacked containers. Since we were demonstrating a proof of concept and the sensors in use for this test were commercial-off-the-shelf devices with no support for multihop communications, three sensors were placed on containers on three of the five railcars nearest to the locomotive so that they could be within radio range of the seal interrogation transceiver. One sensor was placed on the front of the locomotive while the fifth sensor was kept in the locomotive and manually opened and closed while the train was in motion to create events.

During the experiment the VNOC reported events to decision makers using e-mail and SMS messages. The e-mail messages also include a link to Google Maps, so that the exact location of the incident could be visualized. Fig. 9 shows the content of one of the e-mail messages that was sent to the decision makers and Fig. 10 presents an example of an SMS message.

In Figs. 9 and 10, the sensor ID, latitude and longitude data, and event type come from the MRN, while the shipment data comes from the TDE. The VNOC combines these pieces of information into an

```

NOC_AlarmReportingService:
Date-Time: 2009.01.07 07:12:17 CST /
          2009.01.07 13:12:17 UTC
Lat/Lon: 38.83858/-94.56186,
          Quality: Good
http://maps.google.com/maps?q=38.83858,-94.56186
TrainId=ShrtHaul1
Severity: Security
Type: SensorLimitReached
Message: SensorType=Seal
          SensorID=IAHA01054190
          Event=Open Msg=
NOC Host: laredo.ittc.ku.edu

```

```

Shipment Data:
Car Pos: 3
Equipment Id: EDS 10970
BIC Code: ITTC054190
STCC: 2643137

```

Fig. 9. E-mail Message Received During Short-haul Trial

```

NOC_Alarm:
Time:2009.01.07 07:12:49 CST
GPS:38.83860/-94.56186
Trn:ShrtHaul1
Sev:Security
Type:SensorLimitReached
Msg:SensorType=Seal SensorID=IAHA01054190
      Event=Close

```

Fig. 10. SMS Message Received During Short-haul Trial

e-mail message that also includes a link to Google Maps, so that the exact location of the incident can be visualized.

During the test the interrogation transceiver lost communication with the seals for a brief period along the route while the train was stationary and then regained communications once the train started moving.

We believe that this loss of communications was due to electromagnetic interference. However, further investigations are needed to validate this claim.

The short-haul rail trial was a success as all seal events were detected and reported to decision makers using both e-mail and SMS messages. Extensive log files were collected during the test and they were postprocessed to obtain data on TSSN system performance. The results from postprocessing, which are reviewed in Section V, show that the prototype system functioned as expected.

Following this experiment, analysis of event logs obtained from the MRN, VNOC, and TDE revealed that there was a significant amount of clock drift on the TSSN Collector Node during this relatively short trial. The time recorded at the VNOC for receipt of a message, in some cases, was earlier than the time recorded at the TSSN Collector Node for sending the message. Since time at the VNOC is controlled by a Network Time Protocol (NTP) [24] server, we conclude that the clock drift is occurring on the TSSN Collector Node. The clock drift problem was resolved in the next version of the TSSN by using a high performance GPS receiver to get high quality local time. Pulse per second (PPS) output from the GPS receiver was used as an input to the NTP server running on the TSSN collector node. It should be noted that in spite of the clock drift in the TSSN collector node we were able to correct for it in our data analysis by assuming that data from different parts of the TSSN is independent, e.g., the time taken to break a seal and generate an alert message is independent of the time taken to transfer a message from the MRN to the VNOC. As a result we can measure elapsed time in different epochs separately and characterize performance of the TSSN prototype.

IV. POSTPROCESSING OF EXPERIMENTAL DATA

In this section we discuss the framework for postprocessing the results of our experiments. During the short-haul rail trial we recorded events in log files at the geographically distributed VNOC, MRN, and TDE. These log files contained data on message sizes, timestamps, event type, message type (incoming/outgoing) amongst other data elements. Our objective was to postprocess these files to evaluate the performance of the prototype TSSN.

Postprocessing of log files was accomplished using a Java library (LogParser) that was developed in-house. First, the library read in all available information in each log file including time, message size, from and to addresses, as well as the original SOAP message. Information from the MRN, VNOC, and TDE log files in this experiment was combined into a single collection of log entries. We expect that every message transmitted in the TSSN should result in at least two log entries—a transmit log entry (at the originating entity) and a received log entry (at the receiving entity). The LogParser library identified

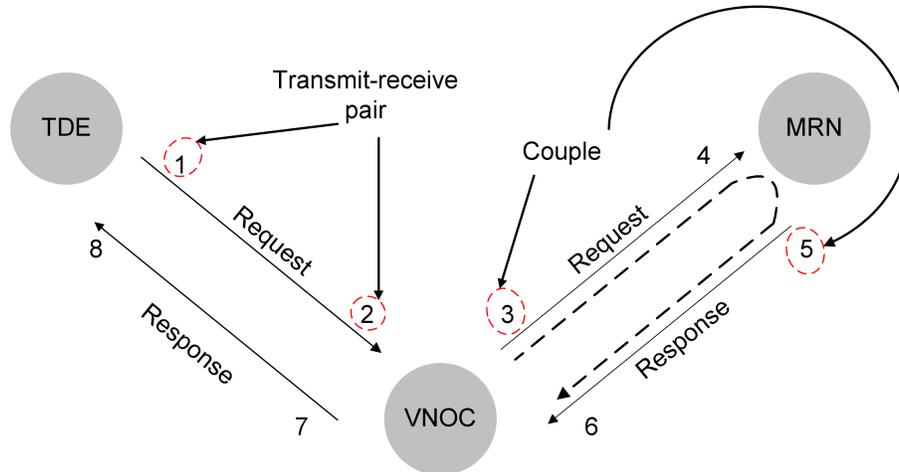


Fig. 11. LogParser Framework Showing Message Couples and Transmit-receive Pairs

log entries as:

- Transmit-receive pairs, that is, the outgoing and incoming log entries with the same SOAP WS-Addressing [13].
- Couples, that is, SOAP request-response message pairs.

Fig. 11 shows the relationship between log entry couples and transmit-receive pairs. Suppose the TDE sends a message to the VNO requesting the current MRN location. The circled “1” and “2” in Fig. 11 denote the log entries representing message transmission from the TDE and receipt of this same message at the VNO. Much of the communication between client/server is based on a request-response model. As a result, there are two related messages which contain additional information to establish their relationship:

- 1) REQUEST: from client to server asking for something; and
- 2) RESPONSE: from server back to the client with the response.

Log entry couples are marked by the records for the outgoing request and response messages. Consequently, the circled “3” and “5” in Fig. 11 constitute the log entry couple for the VNO forwarding the location request message to the MRN and the MRN’s origination of a response respectively. Using the receive pairs for records “3” and “5”, we can also identify entries “4” and “6.”

With this framework, programs were written against the log entry collection to extract the number of messages sent by each service, request-response time for messages, processing time at either the MRN, VNO, or TDE, the time that messages were carried by the network, and message sizes. Additional

information, such as, latitude, longitude, sensor IDs, and event timestamps, is extracted from the SOAP message using XPath expressions. XML Path language (XPath) is used for extracting information from XML by using path expressions that traverse the XML tree. Since SOAP is XML and the elements that we use, e.g., *Alerts*, *MRN_Alarms*, and *VNOC_Alarms*, are also XML, the use of XPath is appropriate. XPath also provides “basic facilities for manipulation of strings, numbers and booleans” [25].

V. RESULTS

This section discusses the results of the experiments presented in Section III. Most of the results shown here are based on the short-haul rail trial because we had more data to analyze. The results presented here are selected to test claims that:

- All messages between component services of the TSSN were transmitted, received, and processed as expected.
- Decision makers can be notified of events on the train in a timely manner.

The rest of this section is laid out as follows: Sections V-A and V-B present results on message counts for the road test and short-haul rail trial respectively. These results test the claim that all messages between component services of the TSSN are transmitted, received, and processed as expected. The rest of the results are based on the short-haul rail trial. Sections V-D–V-E study different portions of the time from event occurrence to decision maker notification to verify the claim that the TSSN can notify decision makers of events in a timely manner. Probability distributions are used in Section V-F to determine the likelihood of timely decision maker notification.

Note that due to significant clock drift in the TSSN collector node, we can only present an estimate of the time taken for an event report to travel from the MRN to the VNOC. However, observed time values can be directly used for other TSSN component interactions. These results show how the aggregate time from event detection to decision maker notification is distributed among the various services and communication links in the TSSN. With this information we will be able to guide system refinements to further reduce the overall time. Suppose that T_n indicates when log entry n is made, then we can compute the following metrics:

- **Service request processing time.** This is the time between when a service receives a request and when a response message is composed. Using Fig. 11, this time is: $T_5 - T_4$.
- **Request-response time.** This is the time taken to get a response from a remote service, including the processing time. Using Fig. 11, this time is: $T_6 - T_3$.

- **Network time.** This is the time taken to get a response from a remote service, excluding the processing time. Using Fig. 11, this time is computed as: $T_6 - T_3 - (T_5 - T_4)$.

Our time analysis in Section V-G examines request-response messages from $\text{VNOC} \rightarrow \text{MRN} \rightarrow \text{VNOC}$, $\text{TDE} \rightarrow \text{VNOC} \rightarrow \text{TDE}$, and $\text{VNOC} \rightarrow \text{TDE} \rightarrow \text{VNOC}$.

The last objective of the short-haul rail trial was to collect data that will be used in a model [26] to support the future design of systems for monitoring rail-borne cargo and determine trade-offs. Message sizes are one component of this model. As a result, Section V-H presents a table summarizing the message size statistics between different components of the TSSN. It should be noted that message sizes can be computed *a priori*; however, the distribution of these messages cannot be determined beforehand.

A. Road Test: Message Counts

The primary goal of the road test was to validate TSSN prototype operation and to determine if correct information is reported by the TSSN collector node, including valid GPS coordinates. During the road test a manual record was made of all seal events and this written record was compared with the information generated from the TSSN. This comparison revealed that all open and close events were propagated correctly. During the approximately 1.5 hours long road test 76 messages (72 *Alarms*, 2 *StartMonitorSensors*, and 2 *StopMonitorSensors* commands) were exchanged on the VNOC to MRN link and these messages corresponded with the events that were recorded in the experiment log. Based on analysis of these messages we conclude that the system operated as expected. In addition, the experiment revealed that the TSSN was able to recover from a dropped HSDPA connection. However, it is worth noting that the seal interrogation transceiver was unable to read the sensors when the trucks were over 400 m apart on a hilly road. Based on the road test we conclude that the TSSN prototype worked as expected in a mobile scenario; we were able to combine sensor data from the MRN in a moving vehicle with shipment information obtained from the TDE to generate e-mail messages that were sent to distributed decision makers. Results from the road test showed that the TSSN prototype was ready for evaluation in a real rail environment.

B. Short-haul Trial: Message Counts

One objective of our postprocessing was to determine if messages were being passed correctly between the TSSN components. During the short-haul trial 203 messages (2 *StartMonitorSensors*, 2 *StopMonitorSensors*, 4 *SensorNodeStatus*, and 4 *SetMonitoringState* commands, 30 *getLocation* queries, 30 *Location* responses, and 131 *MRN_Alarms*.) were passed over the VNOC to MRN link. Full details on the messages

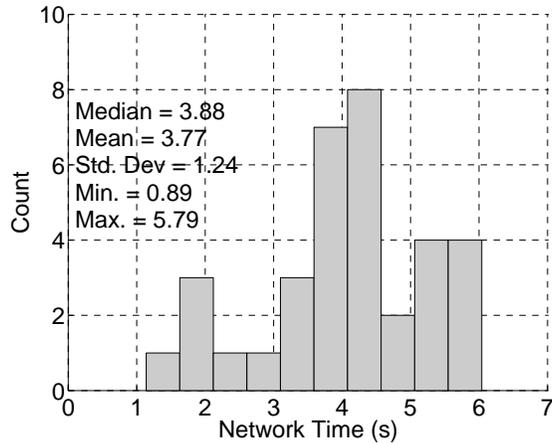


Fig. 12. Network Times from VNOC \rightarrow MRN \rightarrow VNOC

exchanged are found in [27]. All of the *MRN_Alarms* received by the VNOC AlarmProcessor met the necessary rules so that they could be forwarded to decision makers as SMS and/or e-mail messages. The test users who were designated to receive all event notifications from the TSSN received 131 e-mail messages each.

C. Network Time from VNOC to MRN to VNOC

The network time statistics from VNOC to MRN to VNOC allow us to draw conclusions on the time taken to transfer request and response messages from the VNOC to the MRN and *vice versa*. These statistics also allow us to gain insight into the one-way network delay from the TSSN collector node on the train to the VNOC in Lawrence, Kansas—a delay that is one component of sending an *MRN_Alarm* message—which indicates an event at a sensor—from the MRN to the VNOC. Due to clock drift in the TSSN collector node, we are unable to obtain statistics on the one-way network delay from MRN \rightarrow VNOC. However, it is reasonable to assume that the MRN \leftrightarrow VNOC links are symmetric thus, the average one-way delay from the MRN to the VNOC is approximately 1.89 s.

D. Elapsed Time from Alert Generation to AlarmReporting Service

Demonstrating that the elapsed time from alert generation to the AlarmReporting service is of the order of several seconds can help establish the value of the TSSN. Fig. 13 shows the messages involved in notifying a decision maker of an event at a seal. This subsection deals with epochs 2, 3, and 4. Exact values can be computed for the time taken to propagate *Alert* and *VNOC_Alarm* messages, while we can

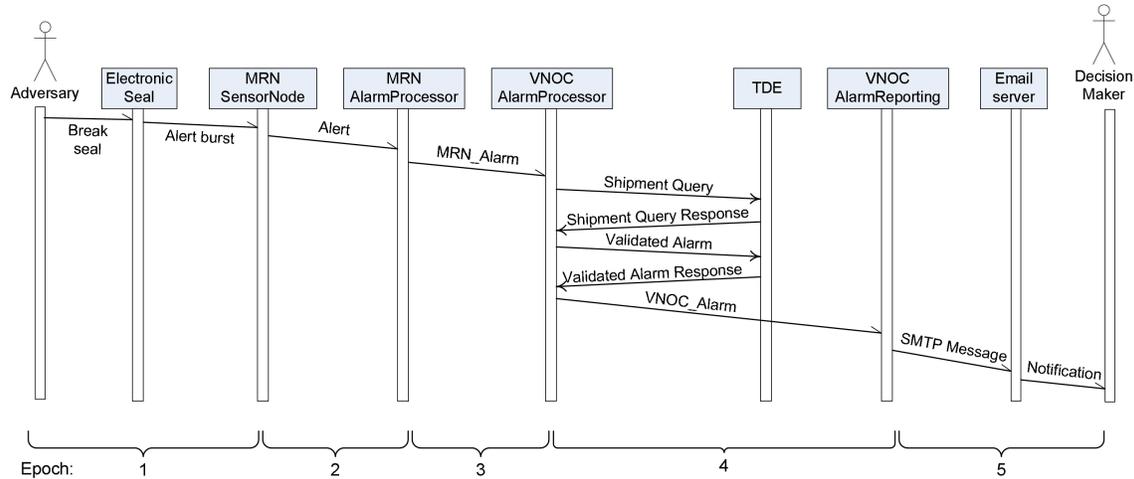


Fig. 13. Sequence Diagram with Messages Involved in Decision Maker Notification

TABLE I
SUMMARY OF TIME STATISTICS FOR DECISION MAKER NOTIFICATION

Epoch	Description	Min./s	Max./s	Mean/s	Median/s	Std. Dev./s
1	Event occurrence to <i>Alert</i> generation	0.81	8.75	2.70	2.13	1.86
2	<i>Alert</i> generation to MRN AlarmProcessor service	0.01	0.08	0.02	0.01	0.01
3	One-way delay from MRN AlarmProcessor to VNOc AlarmProcessor	0.45	2.90	1.89	1.94	0.62
4	<i>MRN_Alarm</i> arrival at VNOc AlarmProcessor to AlarmReporting service	0.01	3.01	0.17	0.05	0.32
5	Elapsed time from VNOc AlarmReporting service to mobile phone	5.2	58.7	11.9	9.8	7.4

use the 1.89 s estimate from the previous subsection as a reasonable value for the time taken to transfer a *MRN_Alarm* message from the MRN to the VNOc.

By analyzing the log files we see that on average it takes about 2 s for messages to get from the MRN SensorNode service to the VNOc AlarmReporting service. Thus, we conclude that the time taken to process events in the TSSN is not an impediment to timely notification of decision makers.

E. End-to-end Time from Event Occurrence to Decision Maker Notification

The primary performance metric for prototype TSSN performance is the time between event occurrence until a decision maker is notified using an SMS message. The components of the end-to-end time include

epochs 1–5 in Fig. 13.

To obtain an understanding of the end-to-end system time as well as overcome any clock errors in the MRN subsystem, we set up a laboratory experiment to determine the elapsed time between event occurrence and the TSSN’s generation of the related event alert. In this experiment, a stopwatch was started when a seal was either broken or closed; when the MRN SensorNode service generated an *Alert* message the stopwatch was stopped. From Table I we see that the longest observed time in epoch 1 is about 8.8 s, while the mean is about 2.7 s.

Since the commercial wireless networks used for decision maker notification are outside TSSN control, a second laboratory experiment was carried out to determine the elapsed time in epoch 5. In this experiment a client program was written to send messages to the VNOC AlarmReporting service. A stopwatch was started when the VNOC sent an alarm to a decision maker and the stopwatch was stopped when the decision maker’s phone received an SMS message. This experiment was repeated for four different carriers resulting in the data shown in row 5 of Table I.

From Table I we see that even though SMS was not designed as a real-time system, it provides excellent notification for this application, since most of our messages were delivered within one minute. Combining all of these results, we see that in these experiments the longest observed end-to-end system time was just over one minute² to notify decision makers of events. Most of this time is spent delivering an SMS message to the decision maker, so we conclude that the TSSN provides a mechanism for timely notification of decision makers.

F. Modeling of Decision Maker Notification Time

In this section we determine the likelihood of timely event notification. To determine the likelihood of timely event notification a probabilistic model is needed for the time epochs shown in Fig. 13. The observed histograms for each epoch visually resembled a Gamma distribution. Thus, in this analysis we assume the times in each epoch followed a Gamma probability density function. While the number of observations (less than 130) was insufficient to statistically validate this assumption this postulate allows us to probabilistically determine if the TSSN prototype can provide event notification within 15 minutes [5] as required. The parameters for the distributions are estimated from the collected data and shown in Table II, where $\hat{\alpha}$ and $\hat{\theta}$ represent the shape and scale parameters of the associated Gamma random

²This time is broken out as follows: in the longest observed times in our experiments it took approximately 8.8 s between event occurrence and the TSSN generating an alert; 2) it took approximately 4.91 s for an alert message to go through the TSSN until notification was sent to decision makers; and 3) it took up to 58.7 s to deliver an SMS message to decision makers.

TABLE II
ESTIMATED GAMMA DISTRIBUTION PARAMETERS FOR TIME TAKEN BETWEEN SEAL EVENTS AND DECISION MAKER
NOTIFICATION

Epoch	Symbol	Description	$\hat{\alpha}$	$\hat{\theta}$
1	E_1	Event occurrence to <i>Alert</i> generation	4.01	0.60
2 + 4	$E_{2,4}$	<i>Alert</i> generation to MRN AlarmProcessor and <i>MRN_Alarm</i> arrival at VNOC AlarmProcessor to AlarmReporting service	1.13	0.13
3	E_3	One-way delay from MRN AlarmProcessor to VNOC Alarm-Processor	13.95	0.14
5	E_5	Elapsed time from VNOC AlarmReporting service to mobile phone	10.44	1.00

variable. Let τ , which is composed of each of the epochs presented in Sections V-C–V-E, represent the total time taken from event occurrence on the train to decision maker notification on a mobile phone. Then $\tau = E_1 + E_{2,4} + E_3 + E_5$ and we use the results from [28] to show that $\Pr[\tau \leq 240 \text{ sec}] = 99.9\%$. These results indicate that the prototype TSSN can notify decision makers in a timely manner with very high probability.

G. Timing Analysis of Other TSSN Interactions

Table III summarizes request/response, processing, and network time statistics for interaction between various TSSN components. The statistics on VNOC \rightarrow MRN \rightarrow VNOC interaction allow us to draw conclusions on request-response and processing times for certain (Start or stop monitoring at the MRN and get current MRN location.) VNOC commands. TDE \rightarrow VNOC \rightarrow TDE interaction statistics give us insight into the time taken to initiate and process commands to start or stop monitoring at the MRN, get the MRN's current location, or to process the setAlarmSecure command. The VNOC forwards these commands to the MRN and returns the MRN response to the TDE. To the TDE, all the elapsed time from when the VNOC receives a message from the TDE until the VNOC sends a response is processing time at the VNOC, even though part of that time is spent forwarding a response to the MRN and waiting for a response. Finally, the statistics on VNOC \rightarrow TDE \rightarrow VNOC interactions allow us to draw conclusions on request-response, processing, and network times for the TDE to store alarm messages and execute shipment queries. Both of these actions are carried out when the VNOC AlarmProcessor service is about to send an alarm to the VNOC AlarmReporting service. Note that there are no results for the MRN to VNOC to MRN interaction. This is for two reasons: first, clock drift in the MRN prevents us from

TABLE III
SUMMARY OF TIME STATISTICS FOR OTHER TSSN INTERACTIONS

Description	Min./s	Max./s	Mean/s	Median/s	Std. Dev./s
Request-response times from VNOC → MRN → VNOC	0.90	10.96	4.39	3.95	2.40
Network times from VNOC → MRN → VNOC	0.89	5.79	3.77	3.88	1.24
Processing times from VNOC → MRN → VNOC	0.00	5.21	0.61	0.01	1.69
Request-response times from TDE → VNOC → TDE	0.34	11.03	4.29	3.94	2.51
Network times from TDE → VNOC → TDE	0.00	4.00	0.14	0.04	0.64
Processing times from TDE → VNOC → TDE	0.29	10.98	4.15	3.85	2.45
Request-response times from VNOC → TDE → VNOC	0.02	0.41	0.12	0.07	0.11
Network times from VNOC → TDE → VNOC	0.01	0.08	0.05	0.07	0.02
Processing times from VNOC → TDE → VNOC	0.01	0.38	0.07	0.01	0.10

TABLE IV
SUMMARY OF MESSAGE SIZE STATISTICS

Description	Min./bytes	Max./bytes	Mean/bytes	Median/bytes	Std. Dev./bytes
TDE → VNOC	846	1278	874.7	848	96.8
VNOC → TDE	968	975	971.5	971	2.6
VNOC → MRN	650	1036	690.8	650	101.5
MRN → VNOC	799	1560	1419.2	1536	237.1

computing a one-way network delay. Secondly, the MRN only generates response messages. As expected there are no request messages originating from the MRN that could be used in a log entry couple to calculate request-response or processing times.

H. Message Sizes

Table IV summarizes the message size statistics for all the messages exchanged in the TSSN. Message size data are needed for a model [26] that is under development to determine system trade-offs as well as optimal or near-optimal sensor locations when using a rail-borne cargo monitoring system. The cost of transmitting a message from the train to an operations center is one component of this model. This transmission cost, in turn, depends on the average message length transmitted from the train and the frequency at which these messages are generated.

VI. REFINEMENTS BASED ON EXPERIMENTAL RESULTS

This section proposes refinements to the TSSN based on experimental results. Recall from Section III-B that we have corrected the clock drift problem by using a high performance GPS receiver to get high quality local time on the TSSN collector node. In addition, postprocessing of the log files also indicated that a unique identifier—perhaps composed of a timestamp and counter—is needed in the *Alert*, *MRN_Alarm*, and *NOC_Alarm* messages to trace an *Alert* message through the TSSN. This identifier can also be used in the future to locate *MRN_Alarm* messages that need to be retransmitted to the VNOG following a loss of connectivity. Finally, the identifier can be used to mark previously processed messages so that the VNOG does not process the same message more than once.

Additional TSSN enhancements include:

- Redesigning the MRN hardware so that the TSSN collector node has redundant backhaul communication capabilities, for example, multiple satellite and cellular modems each with a different provider.
- Creating a comprehensive security framework for the TSSN. Ongoing research is addressing this issue [29].
- Enhancing sensor capabilities so that sensors can engage in multihop communications to enable whole-train monitoring.

The desired result of the research presented here is a standards-based open environment for cargo monitoring with low entry barriers to enable broader access by stakeholders while showing a path to commercialization.

VII. RELATED WORK

In this section we provide a brief overview of related research to monitor trains and to secure cargo in motion. In 2005 Edwards *et al.* [30] presented a prototype system to monitor and control various sensors and actuators on a freight train. The prototype uses a Controller Area Network (CAN) bus to collect data from the sensors. The data is then coupled with GPS information and reported to a web server via a CDMA-based transmitter. Edwards *et al.* [30] argue that “on board sensing of mechanical defects enables car owners to track defects and proactively schedule maintenance” at a time and location that makes economic sense.

The Transf-ID system [31], proposed in 2009, uses radio frequency identification (RFID) tags and a service-oriented architecture to track cargo, railcars, and frequently serviced parts. The authors of [31]

argue that use of the Transf-ID system improves rail freight safety since part maintenance schedules are now based on actual use.

In 2007 Lauf and Sauff [32] proposed a security protocol for transmitting information from sensors within a shipping container to a trusted third party. Such a protocol permits tracing liability for cargo theft and/or damage while minimizing the risk that shipping containers can be used for terrorism or shipment of contraband. The protocol was deployed successfully to test hardware; however, additional research is needed to create tamper-resistant units for monitoring container security [32]. Also in 2007, Ruiz-Garcia *et al.* [33] argued that the technology already exists to develop a monitoring system for refrigerated containers. They added that sensor readings and GPS information can be combined to track a shipping container through different stages of the supply chain.

The review of related work presented in this section shows that others have monitored train equipment using a service-oriented architecture as well as developed security protocols for communicating with sensors inside shipping containers. To the best of our knowledge, the TSSN is the first effort that uses sensors and an open service-oriented architecture to monitor freight in motion.

VIII. CONCLUSION

In this paper we have presented results from field trials of the prototype TSSN (Transportation Security Sensor Network). The TSSN is an open system where different vendors can supply different components of the system. Within the TSSN framework we have successfully combined sensor and shipment information to provide event notification to distributed decision makers. This paper has shown results documenting the interactions between the different components of the TSSN. Based on our experiments and evaluations with the prototype the TSSN architecture is viable for monitoring rail-borne cargo. We have successfully demonstrated that alert messages can be sent from a moving train to the VNOC and combined with cargo information that is forwarded to geographically distributed decision makers using either SMS or e-mail. Furthermore, based on the experiments reported here, we are able to detect events and notify decision makers in just over one minute. Thus, we conclude that the TSSN architecture provides a mechanism for timely notification of decision makers. However, additional development and testing is needed before the TSSN architecture can be deployed in production systems.

ACKNOWLEDGMENTS

The authors would like to thank A. Francis reading and commenting on previous versions of this paper. The authors wish to acknowledge Kansas City Southern Railway for their participation in the short-haul

rail trial. We would also like to acknowledge the support of EDS, an HP company, and Kansas City SmartPort, Inc. our partners on this project. Finally, L. Sackman of EDS, an HP company, assisted with the short-haul rail trial.

REFERENCES

- [1] Federal Bureau of Investigation. (2006, July 21) Cargo Theft's High Cost. Headline. Federal Bureau of Investigation. [Online]. Available: http://www.fbi.gov/page2/july06/cargo_theft072106.htm
- [2] European Conference of Ministers of Transport, *Container Transport Security Across Modes*. Paris, France: Organisation for Economic Co-operation and Development, 2005.
- [3] OASIS. (2006, Oct 12) Reference Model for Service Oriented Architecture 1.0. OASIS Standard. [Online]. Available: <http://docs.oasis-open.org/soa-rm/v1.0/soa-rm.pdf>
- [4] KC SmartPort. (2008, Nov 10) Trade Data Exchange—Nothing short of a logistics revolution. Digital magazine. [Online]. Available: <http://www.joc-digital.com/joc/20081110/?pg=29>
- [5] Kansas City Southern Railroad, Private communication, 2007.
- [6] A. Arsanjani *et al.*, “Web Services: Promises and Compromises,” *Queue, ACM*, vol. 1, no. 1, pp. 48–58, Mar 2003.
- [7] H. Saiedian and S. Mulkey, “Performance Evaluation of Eventing Web Services in Real-time Applications,” *Communications Magazine, IEEE*, vol. 46, no. 3, pp. 106–111, Mar 2008.
- [8] J. Brown *et al.*, “SMS: The Short Message Service,” *Computer*, vol. 40, no. 12, pp. 106–110, Dec. 2007.
- [9] D. T. Fokum *et al.*, “Experiences from a Transportation Security Sensor Network Field Trial,” in *Proc. 3rd IEEE Workshop Enabling the Future Service-Oriented Internet: Towards Socially-Aware Networks (EFSOI 2009)*. Honolulu, HI: IEEE, Dec. 2009, pp. 1–6.
- [10] M. Kuehnhausen, “Service Oriented Architecture for Monitoring Cargo in Motion Along Trusted Corridors,” Master’s thesis, University of Kansas, July 2009.
- [11] R. Chinnici *et al.* (2007, June) Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language. W3C Recommendation. W3C. [Online]. Available: <http://www.w3.org/TR/wsd120/>
- [12] M. Gudgin *et al.* (2007, Apr) SOAP Version 1.2 Part 1: Messaging Framework (Second Edition). Member submission. W3C. [Online]. Available: <http://www.w3.org/TR/soap12-part1/>
- [13] D. Box *et al.* (2004, Aug 10) Web Services Addressing (WS-Addressing). Member submission. W3C. [Online]. Available: <http://www.w3.org/Submission/ws-addressing/>
- [14] OASIS. (2004, March) Web Services Security: SOAP Message Security 1.0. OASIS Standard. OASIS. [Online]. Available: <http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf>
- [15] D. Box *et al.* (2006, Mar) Web Services Eventing (WS-Eventing). Member Submission. W3C. [Online]. Available: <http://www.w3.org/Submission/WS-Eventing/>
- [16] The Apache Software Foundation. (2008, Aug 24) Apache Axis2. Project documentation. The Apache Software Foundation. [Online]. Available: <http://ws.apache.org/axis2/>
- [17] D. Mulvey, “HSPA,” *Communications Engineer*, vol. 5, no. 1, pp. 38–41, February-March 2007.
- [18] C. E. Fossa *et al.*, “An overview of the IRIDIUM (R) low Earth orbit (LEO) satellite system,” in *Proc. IEEE 1998 National Aerospace and Electronics Conference, (NAECON 1998)*, Dayton, OH, USA, Jul 1998, pp. 152–159.
- [19] Hi-G-Tek. (2009, Mar 17) Hi-G-Tek—Company. Corporate website. Hi-G-Tek. [Online]. Available: <http://www.higtek.com/>

- [20] The Apache Software Foundation. (2007, Sep 1) Apache log4j. Project documentation. The Apache Software Foundation. [Online]. Available: <http://logging.apache.org/log4j/>
- [21] EsperTech. (2009, Feb 11) Esper – Complex Event Processing. Project documentation. EsperTech. [Online]. Available: <http://esper.codehaus.org/>
- [22] *DataReader and DataSeal : User's Manual*, UM4710, Hi-G-Tek Ltd., 2001, ver. A5.
- [23] Google. (2009, May 6) Google Maps. Web mapping service. [Online]. Available: <http://maps.google.com>
- [24] D. L. Mills, "Internet Time Synchronization: the Network Time Protocol," *Communications, IEEE Transactions on*, vol. 39, no. 10, pp. 1482–1493, Oct 1991.
- [25] J. Clark and S. DeRose. (1999, Nov 16) XML Path Language (XPath). W3C Recommendation. W3C. [Online]. Available: <http://www.w3.org/TR/xpath>
- [26] D. T. Fokum, "Optimal Communications Systems and Network Design for Cargo Monitoring," To appear in Proc. Tenth Workshop Mobile Computing Systems and Applications (HOTMOBILE 2009). Santa Cruz, CA: ACM Press, Feb 2009.
- [27] D. T. Fokum *et al.*, "Experiences from a Transportation Security Sensor Network Field Trial," University of Kansas, Lawrence, KS, ITTC Tech. Rep. ITTC-FY2009-TR-41420-11, June 2009.
- [28] S. Nadarajah, "A Review of Results on Sums of Random Variables," *Acta Applicandae Mathematicae: An International Survey Journal on Applying Mathematics and Mathematical Applications*, vol. 103, no. 2, pp. 131–140, Sep 2008.
- [29] E. Komp *et al.*, "Implementing Web Services: Conflicts Between Security Features and Publish/Subscribe Communication Protocols," University of Kansas, Lawrence, KS, ITTC Tech. Rep. ITTC-FY2010-TR-41420-19, Feb. 2010.
- [30] M. C. Edwards *et al.*, "Improving Freight Rail Safety with on-board Monitoring and Control Systems," in *Proceedings of the 2005 ASME/IEEE Joint Rail Conference*, Pueblo, CO, USA, Mar 2005, pp. 117–122.
- [31] J. Fernandez *et al.*, "Transf-ID: Automatic ID and Data Capture for Rail Freight Asset Management," *Internet Computing, IEEE*, vol. 13, no. 1, pp. 22–30, Jan.-Feb. 2009.
- [32] J. Ove Lauf and H. Sauff, "Secure Lightweight Tunnel for Monitoring Transport Containers," in *Third Int'l Conf. Security and Privacy in Communications Networks (SecureComm 2007)*, Nice, France, Sep 2007, pp. 484–493.
- [33] L. Ruiz-Garcia *et al.*, "Review. Monitoring the intermodal, refrigerated transport of fruit using sensor networks," *Spanish Journal of Agricultural Research*, vol. 5, no. 2, pp. 142–156, 2007.

The University of Kansas



Technical Report

**Transportation Security SensorNet:
A Service Oriented Architecture
for Cargo Monitoring**

Martin Kuehnhausen and Victor S. Frost

ITTC-FY2010-TR-41420-22

April 2010

Project Sponsor:
Oak Ridge National Laboratory
Award Number 4000043403

Copyright © 2010:
The University of Kansas
2335 Irving Hill Road, Lawrence, KS 66045-7559
All rights reserved.

Table of Contents

Table of Contents.....	i
List of Figures.....	i
Abstract.....	1
I. Introduction.....	1
II. Problem Area.....	2
A. Proprietary Solutions.....	2
B. Variety of Open Standards.....	2
C. Service Oriented Architecture.....	3
III. Related Work.....	4
A. Microsoft - An Introduction to Web Service Architecture.....	4
B. Adobe - Service Oriented Architecture.....	4
C. Open Sensor Web Architecture.....	5
D. Electronic Freight Management.....	5
E. Globus - Open Grid Services Architecture.....	5
F. Service Architectures for Distributed Geoprocessing.....	5
G. Web Services Orchestration.....	5
H. Summary.....	6
IV. Proposed Solution.....	6
A. Overview.....	6
B. TSSN Common Namespace.....	10
C. Mobile Rail Network.....	10
D. Virtual Network Operation Center.....	11
E. Trade Data Exchange.....	13
F. Open Geospatial Consortium Specifications.....	13
V. Results.....	13
VI. Conclusion.....	14
VII. Future Work.....	14
Acknowledgment.....	14
References.....	14

List of Figures

Figure 1: Service message overview.....	7
Figure 2: Service cloud.....	7
Figure 3: Service composition.....	8
Figure 4: Mobile Rail Network message overview.....	11
Figure 5: Mobile Rail Network Sensor Node.....	10
Figure 6: Mobile Rail Network Alarm Processor.....	10
Figure 7: Virtual Network Operation Center message overview.....	12
Figure 8: Virtual Network Operation Center Sensor Management.....	11
Figure 9: Virtual Network Operation Center Alarm Processor.....	11
Figure 10: Virtual Network Operation Center Alarm Reporting.....	12
Figure 11: Trade Data Exchange message overview.....	13
Figure 12: Trade Data Exchange Service.....	13

Transportation Security SensorNet: A Service Oriented Architecture for Cargo Monitoring

Martin Kuehnhausen, *Graduate Student Member, IEEE* and Victor S. Frost, *Fellow, IEEE*

Abstract—This paper describes a system architecture for a *Transportation Security SensorNet* (TSSN) that can be used to perform extensive cargo monitoring. It is built as a *Service Oriented Architecture* (SOA) using open *web service* specifications and *Open Geospatial Consortium* (OGC) standards. This allows for compatibility, interoperability and integration with other *web services* and *Geographical Information Systems* (GIS).

The two main capabilities that the TSSN provides are remote sensor management and alarm notification. The architecture and the design of its components are described throughout this paper. Furthermore, the specifications used and the fundamental ideas behind a SOA are explained in detail.

The system was evaluated in real world scenarios during field trials and performed as specified. The alarm notification performance throughout the system, from the initial detection at the *Sensor Node* service to the *Alarm Reporting* service, is on average 2.1 seconds. Location inquiries took 4.4 seconds on average. Note that the majority of the time, around 85% for most of the messages sent, is spent on the transmission of the message while the rest is used on processing inside the *web services*.

Finally the lessons learned are discussed as well as directions for future enhancements to the TSSN, in particular to security, complex management and asynchronous communication.

Index Terms—Telemetry, Transport protocols, Intermittently connected wireless networks, Communication system software, Data communication, Software engineering

I. INTRODUCTION

THE theft and tampering of cargo are common problems in the transportation industry. According to Wolfe [1] the “FBI estimates cargo theft in the U.S. to be \$18 billion” and the Department of Transportation “estimated that the annual cargo loss in the U.S. might be \$20 billion to \$60 billion”. Wolfe [1] also gives good reason to believe that the actual number may be even higher than \$100 billion because of two reasons. First it is assumed that about 60 percent of all thefts go unreported and second the indirect costs associated with a loss are said to be three to five times the direct costs.

With the advances in technology, this problem has evolved into a cat-and-mouse game where thieves constantly try to outsmart the newest cutting edge security systems.

In terms of securing cargo, there are usually two aspects: first ensuring the physical safety of the cargo and second

monitoring and tracking it. The latter especially has become of more interest as of late because many shipments cross national borders and cargo may be handled by a multitude of carriers. All of this leads to a huge demand for tracking and monitoring systems by the cargo owners, carriers, insurance companies, customs and many others.

This paper is part of a series that describe the design, various components and conducted experiments of the TSSN. In particular we focus on the software architecture here and refer to papers that deal with the other parts of the TSSN in the following. [2] gives an overview of the hardware utilized and describes in detail truck trials and a short haul train trial. [3] presents a new and flexible approach to deal with challenges such as intermittent and low-bandwidth communication in mobile monitoring environments and a long haul train trial in Mexico. Furthermore [4] discusses a framework for analyzing and visualizing SOAP messages to overcome the challenges of complexity and disparity that web service monitoring and management approaches face. Security associated with the TSSN and specifically issues that arose when integrating elements from the *Web Services Architecture* (WSA) led by the *World Wide Web Consortium* (W3C), specifically publish/subscribe communication and service security are described in [5].

Here, a framework is introduced which builds on open standards and software components to allow “monitoring cargo in motion along trusted corridors”. The focus lies on the use of a *Service Oriented Architecture* and *Geographical Information System* specifications in order to allow an industry wide adoption of this open framework.

In the following we discuss the problems of proprietary systems, the advantages of open standards and the approach of using a *Service Oriented Architecture* in the transportation industry. We introduce the design and architecture of the framework and explain the individual components as well as the software parts and specifications that are used in the implementation.

The discussion of proprietary systems in contrast to open standards in the following section provides an overview of the challenges that trade and shipping partners face. It explains why it is important to design an open system that is based on standards. Some of the main advantages are a decrease in cost, more efficient shipment management, and enhanced visibility and tracking capabilities. This paper presents the architecture of the TSSN that was implemented to show that such an open system can be built and deployed successfully.

M. Kuehnhausen and V. S. Frost are with the Information and Telecommunication Technology Center, The University of Kansas, Lawrence, KS, 66045, USA; Corresponding author: mkuehnha@itc.ku.edu

This work was supported in part by Oak Ridge National Laboratory (ORNL)—Award Number 4000043403. This material is also partially based upon work supported while V. S. Frost was serving at the National Science Foundation.

II. PROBLEM AREA

In order to address the problem of cargo security, the *Transportation Security SensorNet* project has been created. Its goal is to promote the use of open standards and specifications in combination with web services to provide cargo monitoring capabilities. The main question is the following:

“How can a *Service Oriented Architecture*, open standards and specifications be used to overcome the problems of proprietary systems that are currently in place and provide a reusable framework that can be implemented across the entire transportation industry?”

The three main aspects of this question are discussed next.

A. Proprietary Solutions

Current commercial systems in the transportation industry are often proprietary. This is because a lot of effort is spent on research and development in order to create *intellectual property*. The assumption is then that as long as the competitors do not have access to the system and its protocols that *intellectual property* is safe and provides a competitive advantage. Another common “benefit” of keeping the systems closed is the perceived additional security since in order to successfully attack the system its implementation and protocols have to be *reverse engineered*.

The problem with this is that these advantages are often one-sided and lead to stove pipe systems provided by a single vendor. Once a proprietary system has been implemented it has to be maintained. What happens if a customer that uses the system invested a lot of money into a its infrastructure and the training of its employees and the company that provides the system releases a new version of it which of course costs money again. The customer has several choices:

1) *Upgrade*: Throughout the literature this is often considered the most expensive option because of the cost for the upgrade to the new version and the additional training to the employees that has to be provided. The benefits of upgrading are the use of new technology, potential gains in efficiency through new features and the latest bug fixes.

2) *Do Not Upgrade*: By many regarded as the most cost efficient solution, choosing not to upgrade compromises new features and updates for the ability to save costs. An approach that is taken by some companies is the *skip a version* technique. This allows companies to plan better as internal processes and systems often have to interoperate and need to remain compatible to each other.

3) *Change Vendor*: In this situation, the new version of the system that is provided by company A does not provide the necessary features or is simply too expensive. Furthermore, a different company B offers a similar product with more features or for less money. The move to the new system is now dependent on the following things: How big are the estimated savings and what are the direct and indirect costs of the transition? It often happens that after careful consideration the costs outweigh the estimated gains and the customer goes back to considering whether or not to simply upgrade. If a

transition is made, the process could be time consuming and turn out to be more complicated than expected.

Picture this extreme case as well. What happens if the vendor goes out of business? All of the sudden, the short-term goal is to maintain support for the system and to keep it running while in the long-term to look for a suitable replacement and be forced to transition. Even if this case does not happen the dependency on the vendor can be crucial. If the system has errors or a particular enhancement is desperately needed, the vendor decides what to do about it. For big companies that are major customers this may not be such a big problem because they often get preferential treatment. But for small and medium businesses the wait might be too long and lose them customers and revenue.

The main point here is that many customers are locked into proprietary solutions that are incompatible with similar solutions offered by competitors. In a 2003 survey by the Delphi Group [6] it was found that 52% of developers and 42% of consumers see standards enabling the “approval of projects otherwise threatened by concerns over proprietary system lock-in”. Furthermore, an overwhelming 71% of developers and 65% of consumers feel that the use of open standards “increases the value of existing and future investments in information systems”.

The problem of non-interoperability with regard to geospatial processing is the topic of a paper by Reichard [7]. Because *Geographical Information Systems* are often immensely complex, companies that invest heavily into this area often only support their product. As described in the sample scenario, this leads to a lack of coordination among entities such as the *Federal Emergency Management Agency* (FEMA), the *National Transportation Safety Board* (NTSB) and the *Environmental Protection Agency* (EPA) because of the inability to share vital information which is the key to fast decision making and data analysis

B. Variety of Open Standards

1) *Standards principles*: The idea of open standards and specifications is to define *interfaces* and *protocols* that can be used as references for the implementation of a system. There are many standards committees and industry groups that aim to define them, most often focused on a particular area. Some of the most well-known ones include the *World Wide Web consortium* (W3C), the *Organization for the Advancement of Structured Information Standards* (OASIS), the *International Telecommunication Union* (ITU) and the *International Organization for Standardization* (ISO).

The main principles that govern the development of standards are usually the same across all organizations. The following is an overview according to ISO:

a) *Consensus*: All parties that are affected by the proposed standard get the chance to voice their opinions. This includes initial ideas and continues with feedback and comments during the standardization process.

b) *Industrywide*: The idea is to develop global standards that can be used worldwide by entire industries.

c) *Voluntary*: The standardization process is driven by the people that are interested in it and that see its future benefits across a particular industry. It is often based on *best practices* that are already commonly in use.

2) *Aspects of Open Standards*: The importance of open standards is emphasized in a paper by McKee [8]. It provides the evolution and success of the Internet as the “perfect example” for the use of open standards. In particular it explains that since the Internet is based upon communication and communication means “transmitting or exchanging through a *common system* of symbols, signs or behavior”, the process of standardization can basically be seen as “agreeing on a *common system*”. The other parts of the paper are focused on how *openness* can help *Geographical Information Systems* (GIS) but many of the points mentioned apply to open standards in general.

In particular the following aspects are associated with open standards:

a) *Compatibility*: This includes the ability to share data across vendors and systems in a uniform and non-proprietary form. It allows processes to use essentially the same data in order to perform their specific task without the need of costly conversions or interpretation errors. Most *common* formats are also backward compatible which means that no particular version of the system is needed to interpret the data. Only a certain subset of functionality might be provided when using in older versions though. Another advantage of open formats is the fact that even if a particular version of a format is completely outdated and only used in legacy systems, its specification is still accessible to everyone. Hence systems can still be designed to use the format.

b) *Freedom of Choice*: A major problem of proprietary solutions that was described earlier was the *vendor lock*. Once a customer implements a proprietary system and builds its infrastructure around it, choices in the future are limited. Open standards by definition are vendor independent. Furthermore many of them support a broad variety of implementation scenarios. These implementations often are not even limited to a particular platform, operation system or programming language. This is especially true for most of the web standards.

c) *Interoperability*: Through the use of clearly defined interfaces, standards dramatically enhance interoperability. The standards that define interface specifications do not provide a specific implementation but provide references to *best practices* and *implementation patterns* instead. Companies choose what kind of system implementation they prefer. This allows them to make use of existing infrastructure and capabilities that might otherwise have to be changed when using a proprietary system. The uniform access to functionality and data enables companies to connect a multitude of systems and make more use of them. Also, in case one part of the system has to be replaced, another one that simply provides the same interface can take its place. This allows great flexibility in terms of the overall system design.

d) *Leverage*: For companies the standardization of concepts, frameworks and common approaches provides a number of benefits. Since research and development can be extremely cost intensive, companies want to make sure there is a guar-

anteed *return on investment* for them. Open standards do not necessarily lead to increased revenue but they do provide insurance to the companies that they are on the “right” track and what they implement is actually used industrywide. This is very important because customers are aware that when they purchase a system from company A that uses a proprietary or non-standard implementation they might become a victim of *vendor lock*. Acquiring a system that is build on open standards allows them to choose the best and most cost effective solution from a variety of independent implementations. Another advantage is that once different implementations by the main vendors have been established, there is room for custom solutions by smaller vendors, often in the form of extensions or plugins.

e) *Open Source*: The biggest benefit of using open standards is that fact it leads to innovation. This is because everybody can contribute, suggest enhancements, outline *best practices* and address mistakes. In terms of software this approach is often referred to as *open source*.

However, there are several problems that can be associated with non-proprietary systems. Implementations are based upon the interpretation of the standards which may differ significantly. Furthermore, some implementations only support a subset of the original specification, are slower than the reference implementation or use incompatible sub systems.

C. Service Oriented Architecture

The concept of information processing and sharing across various applications using *web services* is the main focus of this paper. The basic idea is to define components of a system as *services* and users as *clients* that can retrieve data from them. Note that interaction between *services* is done using *embedded clients*. The *services* take care of things such as information processing, data analysis and storage. With all business logic embedded into *services* and interaction between them clearly defined using open standards an infrastructure is built that is called the *Service Oriented Architecture* (SOA).

The Internet allows the following two things that are relevant to information processing: a common means of communication and the ability for efficient information sharing. There exist many standards on how to transmit, receive, encode and decode data. SOA builds on top of them to provide new specifications that enable the design, implementation and use of *web services*. Through these *web services* companies, government agencies and others have the ability to share and process information in a uniform manner which cuts costs, time and resources and improves efficiency.

Now why is SOA such an “enabler”? What is possible now that was not possible before? According to Irmén [9] *automation* and *efficient communication* with partners are the two most important things in *supply chain management* which represents the core of the transportation industry. Let us take a look at how the *Service Oriented Architecture* addresses both of them in regard to the individual topics outlined in [9].

1) *Automation*: A vital part in transportation is the *screening* process. Companies that transport goods must ensure safety and therefore check all parties involved in the trade.

An important aspect of this is the use of a *denied trade list* which lists items and companies that are not allowed to import or export into specific countries. With the reduction in manual labor and transition to a *web services* based system that automatically performs these checks, efficiency could be greatly increased.

A closely related topic is *accountability*. Who is responsible if something goes wrong during the trade process? Since goods are often handled by many different parties, it must be possible to monitor the location of cargo and handovers tightly. This is especially important in cases of tampering or even theft of the cargo.

Furthermore, agencies and customs more and more require *electronic trade information* instead of paper documents in order to track trade. Because of different formats and legacy applications that are often unable to provide this information in its entirety, additional resources have to be allocated in order to remain compliant with current practices. *Web services* and open standards can overcome this problem with uniform interfaces and common data formats.

Having the ability to *monitor* the location not just for perishable goods but also for high value goods is of great importance in the transport chain. Current processes should be able to automatically route cargo based on its needs and cost effectiveness.

Irmen [9] also points out that “the lack of integration between products causes users to deal with multiple systems having disparate data and non-uniform input and output” and calls for the use of a single platform. Using the *Service Oriented Architecture* this “call” becomes less necessary because it is platform independent and at the same time able to provide integration of multiple systems and standardized data formats.

2) *Efficient Communication*: Building a virtual network among the parties involved in the trade process establishes efficient means of communication. It allows the *coordination* between otherwise disparate entities that is essential to provide cost effective and reliable shipping of cargo. The Internet provides the communication layer but it is the standards of *web services* that enable the integration of different systems.

Irmen [9] mentions the *Software-as-a-Service* (SaaS) approach which allows software to run on a per-use basis without the costs of complex hardware infrastructure. This works very well with SOA as the interfaces defined by those services are often *web services* interfaces that are essentially part of SOA.

Security within the transportation industry plays a big role because trade data is to be kept confidential and only distributed on a need-to-know basis. This puts an additional burden on the parties that are involved, as the parties must exchange data confidentially at each point of interaction. If open standards are used for this, *security* is implemented based on interfaces and policies that are easy to manage.

In order to manage the transportation chain in its entirety, a *global view* is often needed. This is problematic since individual parties often only deal with their respective neighbors. Using open standards and the *Service Oriented Architecture* approach each party could provide an uniform information interface that is accessible to other parties in the chain. This allows consistent reporting, monitoring and analysis at each

step during the shipping process.

The reporting part especially has gained more attention over the past years as the focus has shifted towards more ethical and socially responsible business practices. *Accountability* coincides with this *social visibility* and therefore improvements in monitoring cargo not only lead to increased revenue on the business side but better public relations as well.

Overall the paper by Irmen [9] gives excellent reasons for open systems in terms of accountability, coordination, scalability and cost, these important aspects that need to be taken into consideration when designing an architecture such as the *Transportation Security SensorNet*.

III. RELATED WORK

In the following sections related work that is relevant to various aspects of the *Transportation Security SensorNet* such as *Service Oriented Architecture*, web services, communication models, the *Open Geospatial Consortium* specifications and sensor networks is analyzed.

A. Microsoft - An Introduction to Web Service Architecture

Cabrera et al. [10] outline concepts that led to the implementation of *Service Oriented Architectures* and development of the *web services* specifications that surround them and are used by the TSSN. A lot of the main approaches have been standardized in various committees and organizations by now but were only in the early stages when Cabrera et al. first discussed them.

B. Adobe - Service Oriented Architecture

An Adobe technical paper by Nickul et al. [11] outlines general architecture approaches that can be taken when transitioning business processes to the *Service Oriented Architecture*. It mentions a widely used technology called the *Enterprise Service Bus* (ESB) that provides a standardized means of communication for all services that connect to it. For the TSSN this is of importance when it comes to asynchronous communication as the *Java Message Service* (JMS) uses queues that are on the ESB for message exchanges (see [3] reference, IV-A6).

In addition to the basic *Request-Response*, several other *message exchange patterns* that go beyond the standardized ones are described. A *registry* keeps track of service metadata. The *service provider* is responsible for updating it whenever a change occurs and the *service consumer* subscribes to the *registry* for any of these changes. The metadata that is provided is then used to configure a *service client*. Hence, the client can issue *requests* and receive *responses*.

The TSSN essentially uses a very similar approach with the UDDI. Web services automatically register with the UDDI when they are started and clients are able to use specific services by looking them up in the UDDI.

C. Open Sensor Web Architecture

An approach to implement the proposed standards of the *Sensor Web Enablement* (SWE) that are described in [12] is outlined by Chu et al. [13]. A more detailed definition of the system and its core services is provided in the thesis by Chu [14]. The system is called *NICTA Open Sensor Web Architecture* (NOSA) and is focusing on the combination of sensor networks and distributed computing technologies.

The TSSN uses a similar approach but has some significant differences. The goal of both implementations is to integrate a sensor network into a web services architecture using open standards. NOSA uses a sensor application that is tightly integrated into the *Sensor Operating System* and then provides sensor data and control to web services in a non-standard format. TSSN on the other hand implements sensor management and monitoring functionality inside a single service, the *Sensor Node* (see IV-C1) and allows different sensors to be “plugged in”. This allows other services to use standard web service interfaces and SOAP messages in order to access sensors.

Furthermore, the web services used by NOSA are implemented manually according to the OGC specifications which causes them to be limited as not everything that is specified is also implemented. In contrast, the TSSN uses automatic code generation (see IV-A1d) that enables it to use all OGC specifications. Since their elements and interfaces are generated the only thing that has to be implemented is functionality. This approach significantly reduces development efforts.

D. Electronic Freight Management

The Electronic Freight Management (EFM) initiative [15] is a project that focuses on the improvement of communication between supply chain partners using web technologies. One of the main goals is to provide a common and open technology platform for sharing cargo information among smaller and medium size trade partners. The idea is that the information is only entered once and then shared among members of the supply chain.

EFM because of its SOA approach provides a common electronic communication platform that maintains cargo related information on a web service basis. This information is then shared with authorized users while digital certificates and web service security ensure data integrity and confidentiality. The key benefit here is the improved visibility of shipment information which enables all supply chain members to perform their processes more efficiently and plan ahead better [16]. The data exchange is standardized and based on the Universal Business Language (UBL). Furthermore each individual transaction is uniquely identifiable by a Unique Consignment Reference (UCR).

The TSSN approach is similar but deals in particular with cargo monitoring in mobile environments. The Trade Data Exchange (TDE) as described later is responsible for managing and sharing shipment information.

E. Globus - Open Grid Services Architecture

Globus is an architecture that is based on grid computing. It focuses on providing capabilities as services in a grid

environment using standard interfaces and protocols. An initial paper by Foster et al. [17] gives an overview of the architecture and design decisions. In particular, Globus supports “local and remote transparency with respect to service location and invocation” and “protocol negotiation for network flows across organizational boundaries”. Its service approach is similar to the *Service Oriented Architecture* that is used by the *Transportation Security SensorNet*. Additionally, security concepts that work inside a grid are applicable to SOA and vice versa.

The current architecture of Globus is still based on the same principles that were initially described by Foster et al. [17]. The combination of custom components and web services components provides an architecture for security, data management, execution management, information services and a common runtime in a grid environment.

In contrast to the TSSN, Globus makes use of web service specifications in some of its components but also provides custom implementations and interfaces as for service discovery and notifications. The TSSN uses web services specifications and OGC standards almost exclusively which ensures standards compliance and compatibility. For service discovery the UDDI [18], [19] is used and for notifications *WS-Eventing* [20].

F. Service Architectures for Distributed Geoprocessing

A research article by Friis-Christensen et al. [21] outlines the implementation of an application that analyzes the impact of forest fires using web services. The main focus is the transition from a client application to a flexible web services architecture using *Open Geospatial Consortium* specifications. The components include multiple data sources that are made available through data access services like the *Web Map Service* and the *Web Feature Service*. A geoprocessing service performs the analysis of the data and provides it to a client. Furthermore a discovery service serves as the registry for all services and their metadata.

The prototype implemented uses synchronous communication in between services. The problem in this case is that the actual processing can take quite a long time. In the future the authors want to transition to an asynchronous communication model that is similar to the *OGC Web Notification Service*.

In addition, it is pointed out that even though standardized interfaces allow for a combination of services which provides flexibility, the transport of high volumes of data is often not feasible in geoprocessing scenarios which can lead to highly specialized but not very reusable services.

The implementation is interesting in the sense that it exclusively uses OGC specifications which makes it compatible to other *Geographical Information Systems*. The TSSN aims to be OGC compliant as well but includes specifications that deal with sensor networks such as the *Sensor Observation Service* and the *Sensor Alert Service*, something that Friis-Christensen et al. [21] does not address.

G. Web Services Orchestration

The problem of reusability of services and “next generation challenges” was addressed by Kiehle et al. [22]. The idea

here is to increase transparency and reusability by splitting processes into smaller more reusable processes and utilizing a work flow management system called *Web Services Orchestration*. This is especially important for the integration of the *Transportation Security SensorNet* into systems used in the transportation industry. Its modular design and architecture allow single components to be reused and information flows to be created.

The *Web Processing Service* specification describes how services can be arranged and combined into *service chains* that form a process. Two alternatives are commonly used in order to achieve this. A *Web Processing Service* can be set up to combine and “encapsulate” other individual web services and therefore provide the desired abstraction. However, the best way to define work flows is using the *Business Process Execution Language* (BPEL). BPEL enables complex *service chains* to be defined without the need for custom and potentially not reusable *Web Processing Services* that just “encapsulate” services.

H. Summary

The related work addresses the following key technologies that play an important part in the *Transportation Security SensorNet*:

1) *Service Oriented Architecture*: The development of SOA and its web services specifications has come a long way but is still far from over. Even though specifications exist, organizations and businesses often implement components that are similar to the specification but not compliant. As discussed before, this is the case for service discovery and notifications in Globus. Two common reasons behind this are the following. First, the specification may be available but there are hardly any reference implementations that can be used. Second, extensions to the specification that are necessary for a particular implementation or in a specific environment such as the grid are not covered by the standard.

2) *Open Geospatial Consortium*: The OGC specifications are often complex and there is significant development effort necessary to implement the elements, interfaces and functionality they define. Automatic code generation as described IV-A1d and used by the TSSN can facilitate their implementations but is not used very often.

3) *Sensor Networks*: The implications on communication models that sensor networks have, in particular asynchronous message exchanges, are often ignored in web service architectures. As seen in NOSA, the focus is on the implementation of a subset of OGC standards for a particular sensor network, but the link to an overall SOA seems to be missing. It is evident that current systems seem to lack the combination of SOA, OGC specifications and sensor networks. The TSSN combines all these technologies and bridges the gap between implementations that just deal with SOA and OGC specifications and systems that use OGC standards in sensor networks.

IV. PROPOSED SOLUTION

A. Overview

This section describes the architecture of the *Transportation Security SensorNet* (TSSN). It provides an in-depth discussion

of design aspects and the implementation.

1) *Service Oriented Architecture*:

“Service Oriented Architecture (SOA) is a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains.” [23]

Building a “Service Oriented Architecture for Monitoring Cargo in Motion Along Trusted Corridors” makes sense. According to a study by the Delphi Group [6], companies that collaborate usually request compliance for the following standards: XML 74%, J2EE (Java) 44% and SOAP 35%. The architecture used for the implementation of the TSSN utilizes all three technologies by separating functionality into *web services*. This allows for high flexibility and is cost effective.

Haas et al. [24] early on proposed various *models* for web service architectures. The *Message Oriented Model* focuses on message relations and how they are processed. An approach that centers around resources and ownership is the *Resource Oriented Model*. The *Policy Oriented Model* defines constraints and focuses on security and quality of service. Ideas from all these models have been combined with the *Service Oriented Model* into what has become SOA. Of the proposed models it has been the most widely implemented.

A book that provides an excellent overview of Java and *web services* is written by Kalin [25]. Note that SOA by definition is programming language and platform independent. It is built on the basis of requests and responses and the independence of *web services*. The choice to use Java for the implementation was made because the TSSN is built on top of previous research on the *Ambient Computing Environment for SOA* by Searl [26] which is written in Java.

The main components of the TSSN are sensor management and alarm notifications. An overview of the services and relevant message exchanges is shown in Figure 1.

The *Trade Data Exchange* (TDE) (see IV-E) provides shipment, route, logistics and relevant cargo information. It is managed externally and used by the system only through its specified interface. The *Virtual Network Operation Center* (VNOC) (see IV-D) is responsible for the processing of sensor data and alarms. One of the major capabilities that it provides is alarm notification. The *Mobile Rail Network* (MRN) (see IV-C) deals with the actual management of sensors on a mobile platform, e.g. a train. *Web services* at the *Mobile Rail Network* capture sensor data from the sensors and “preprocess” that data. A detailed description of each individual service is provided later in this section.

The architecture consists of web services that are separated into *service clouds*. These *service clouds* represent the different geographically distributed locations (e.g. Overland Park, KS for the TDE; Lawrence, KS for the VNOC and on a moving train for the MRN) where services are deployed and are shown in Figure 2.

The *web services* are developed according to the *web service* specifications and the standards provided by the OGC. This means that they aim to be standards compliant. Since the OGC specifications are at times very complex, the *Geography Markup Language* (GML) for example defines over 1000 elements, the basis for the framework was implemented using

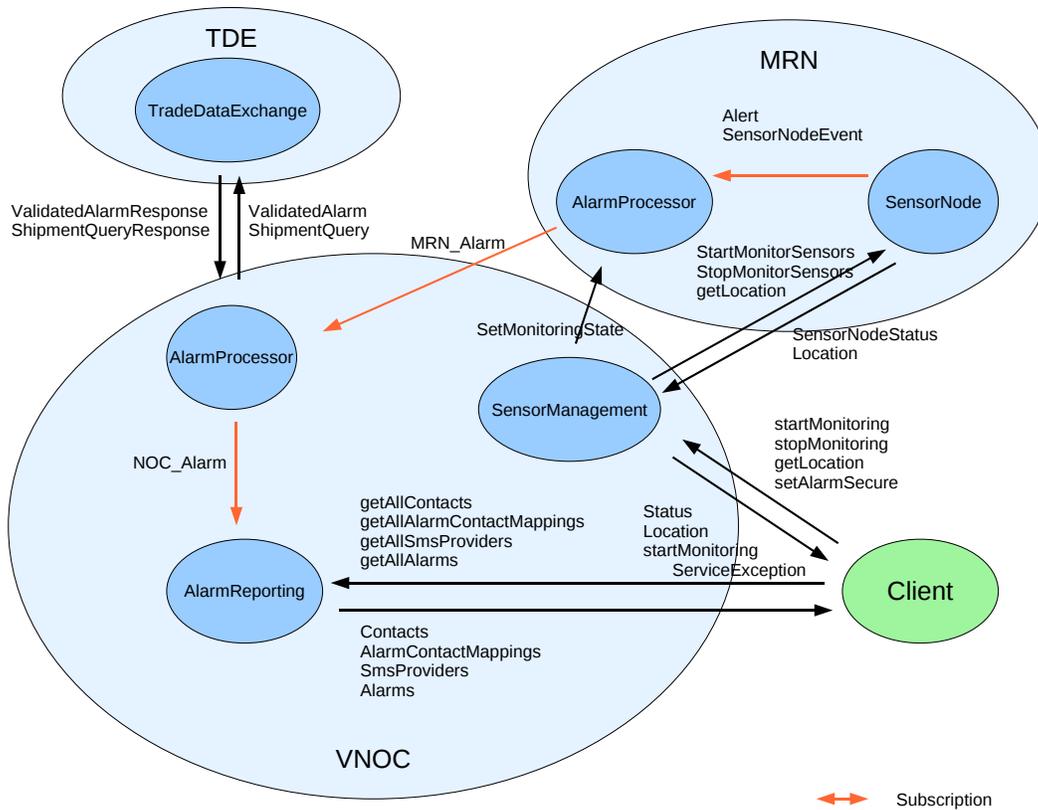


Fig. 1. Service message overview

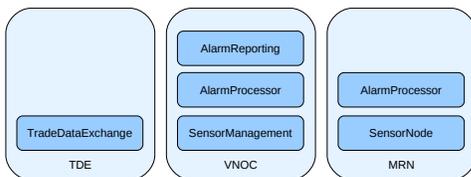


Fig. 2. Service cloud

custom interface definitions first and adding the OGC ones later. This enabled fast prototyping and testing of the system.

The following sections explain in-depth the approaches and technologies used in the architectural prototype and implementation of the TSSN.

a) *Ambient Computing Environment for SOA:* The infrastructure described by Searl [26] called *Ambient Computing Environment for SOA (ACE_SOA)* forms the basis of the implementation of the TSSN. It provides a complete SOAP stack using *Apache Axis2* and a variety of other useful programs that assist in the development of a SOA. ACE_SOA deals with multiple ownerships and federations that provide *web services*. In particular it covers the following aspects:

- *Service Discovery* across different federations
- *Authentication* of clients and services
- *Authorization* of clients and services
- *Subscriptions*

The implementation of the capabilities provided is based on *Apache Axis2* and *web service* specifications. It is explained in detail in the following sections.

b) *Apache Axis2:* *Apache Axis2* is a software stack that allows the development and running of *web services* and clients. Its architecture as described by Chinthaka [27] consists of the following main components:

AXIs Object Model (AXIOM): AXIOM is an XML object model that aims for high performance while requiring low amounts of memory. The idea behind it is the application of a *pull parser*. This allows objects to be built from XML only up to the information that is needed by the user while the rest of it is *deferred*. The advantage of this is that the memory that an object requires is significantly reduced. Furthermore, this approach also increases performance since the entire object model does not have to be constructed before information can be retrieved, which is the case in the Document Object Model (DOM) parser.

Extensible Messaging Engine: Axis2 provides a very modular architecture that allows for a variety of different implementations of *web services* as long as they adhere to certain specifications. A variety of transports such as HTTP, SMTP, JMS and TCP can be used for message exchanges. Inside the *engine* each message goes through *phases* that are part of the *pipng model* which is used to implement *Message Exchange Patterns (MEP)*. Inside these *phases* messages can be modified, filtered or processed. The advantage of doing

this inside a *phase* is that it applies to all messages. This allows for service independent processing implementations. The *message receiver* will then be responsible for handing over the actual message to the service implementation accordingly. They also take care of *synchronous* and *asynchronous* message communication.

Context Model: Axis2 provides a hierarchical context model that distinguishes between the following levels:

- *Configuration of Axis2*
- *Service Group* which is a collection of *services*
- *Service* which contains several *operations*
- *Operation* that consists of *messages*
- *Message* that is sent or received

These contexts are important in the implementation of *web service* specifications such as *WS-Security* and *WS-Policy*. It means that these specifications can be applied on a level basis which provides great flexibility.

Pluggable Modules: In order to provide even more flexibility and to make the implementation of *web service* specifications easier to use, Axis2 provides *modules*. These allow an implementation of message processing that is common and useful for many *web services* to be shared. *Modules* can also be *engaged* or *disengaged* on the following levels:

- *System* which means that every service makes use of the module such as *WS-Addressing*
- *Service* which useful for *WS-Eventing*
- *Operation* that for example allows fine grained security using *WS-Security*

More information about the modules that are used in the TSSN see IV-A4.

Data Binding: Since a majority of data processing, element definitions and interface specifications are in XML, Axis2 provides a variety of *data binding frameworks* such as XMLBeans [28], Java Architecture for XML Binding (JAXB) [29] and JiBX [30]. In addition, the *Axis2 Data Binding (ADB)* can be used, which due to its tight integration with Axis2 is highly performant. For instance, every object contains a *factory* that is able to transform XML into the specific object and vice versa.

Further development was done by the author on this *data binding* to support a full range of OGC specifications such as the *Sensor Observation Service*, *Sensor Alert Service* and most notably the *Geography Markup Language*.

As part of this work several changes to the initial version of Axis2 were made in order to either fix bugs or support more functionality. In particular the build structure was adapted to work better with the TSSN development. It makes extensive use of *Apache Ant* for the automatic generation of elements from their respective XML schema definitions, the compilation of Java classes and the deployment of *web services* and clients.

c) *SOAP: Service Oriented Architectures* make use of SOAP [31] as a flexible message format. The TSSN does the same since *web service* specifications can easily be integrated and applied to SOAP messages.

d) *WSDL:* All services in the *Transportation Security SensorNet* are defined using the *Web Services Description Language (WSDL)* version 2.0. An in-depth introduction is

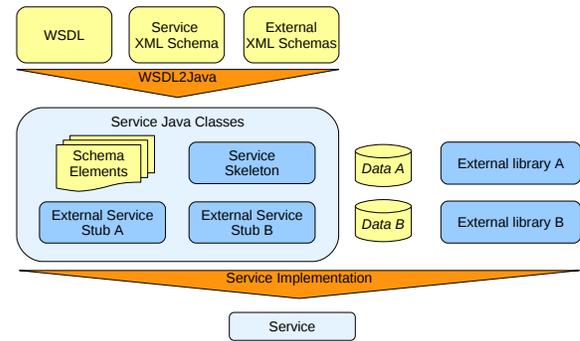


Fig. 3. Service composition

provided in [32]. This section explains how the combination of WSDL files and XML schemas make up the foundation of a web service.

Utilizing the automatic code generator of Axis2 called *WSDL2Java*, all elements defined in the XML schemas are available as Java classes. Furthermore a *skeleton* is created that contains the operations of the web service as methods. Interaction with other services is achieved using their respective *stubs* which provide methods for each of its defined operations. They allow clients to perform requests directly using Java. This is because Axis2 provides the entire SOAP stack from the message format to the parsing into elements all the way up to the invocation of a method that represents a service operation. The composition of the generated parts, data and external libraries then forms the actual service implementation (see Figure 3).

2) *Services:* The services that are implemented in the TSSN make use of a variety of components. For long term information storage, a MySQL database is used. A *object-relational mapping* tool called *Hibernate* [33] enables objects to be stored and retrieved transparently without the need of complicated database interactions. *Esper* [34] provides complex event and alarm processing and is used at the VNOC. The *Alarm Processor* at the MRN currently uses a less complex approach. The *Sensor Node* is responsible for the actual communication with the sensors. It must use a device specific protocol [35] and a serial connection library for Java called *RXTX*.

Each component and its particular use is explained in the later sections when each individual service is described. At a high level, one of the main aspects when dealing with web services is the definition of whether they are *stateless* or *stateful*:

a) *Stateless:* By default web services are meant to be *stateless*. This is because most message exchanges are completely independent of each other. Web services usually offer calculations, information or capabilities that only require the service to perform a specific action and give a response. This is part of the *autonomy* approach of web services.

Even in the case where a web services provides data, the service is still considered *stateless* since the retrieval of the data at any given time is not dependent on the internal state

of the service but only on the underlying data. If the data changes there is no state change in the web service and it still provides the same functionality.

b) *Stateful*: The need for *stateful* web services has been identified for the TSSN because there are certain limitations in just using *stateless* web services. Given a *online* data processor that analyzes sensor data; using a *stateless* web service, it is impossible to react to trends and complex events because the service is limited to single data objects that it receives.

Let us say that a web service is monitoring whether seals that lock cargo containers are broken and is supposed send out warning messages whenever they are. The service has limited capacity in terms of storing historic data but should still be able to intelligently determine if a sensor reading that shows that a seal is broken is just a misreading or a real threat. This is only possible if the service keeps track of previous states. In contrast, a *stateless* service would only be able to react to the current reading and is forced to make decisions based on this single piece of data. Another example is the *Alarm Processor* service (see IV-C2) at the MRN that is used in the TSSN implementation. It classifies sensor data from containers either as *information* or *security* depending on whether one is currently allowed to open the container or not.

3) *Clients*: Clients are able to make use of the operations provided by the *web services*. They usually utilize the same modules as the service. This means that in theory all *web services* could have clients. Since a lot of the services in the TSSN interact independently from users, the number of clients that are available to users is actually smaller.

One of the aspects of clients in the TSSN is the management of the sensors. The *Sensor Management* service (see IV-D1) provides this among other things like retrieving the location of a particular *Sensor Node*. Another aspect is the management of alarm notifications. For this purpose the *Alarm Reporting* service (see Figure 10) defines various management operations for clients.

To facilitate the use of those clients, a *Command Center Graphical User Interface* was implemented that works just like a desktop application. This is in addition to the command line interface that every client provides using the *Apache Commons Command Line Interface (CLI)* library.

4) *Modules*: Axis2 provides the possibility to “plug in” *modules* that add functionality or change the way a service behaves. This allows a specific capability to be shared among different services without having to implement it in each of them. In general, the web service specifications that are used in Axis2 are implemented as modules. For more information see IV-A1b.

a) *Ping*: In order to check the status of a particular service Axis2 provides a module that adds an operation called *pingService* to a service. This can be used to check the status of either a specific operation or all operations that the service defines. The client part that actually uses this operation was not part of Axis2 and had to be implemented by the author.

b) *Logging*: Especially for debugging purposes and performance evaluations, it is of great benefit to be able to see the raw SOAP messages that are sent and received. A *logging* module was implemented to provide this functionality. In

particular the following information is captured of each SOAP message:

- *Time* when the message was sent or received
- *Service* which is used
- *Operation* that is being executed
- *Direction* of the message, which can be either incoming or outgoing. Note that there are special directions that deal with incoming and outgoing faults.
- *From* address of the message
- *Reply to* address that may differ from the *From* address
- *To* address of the message
- *Schema element* that is being “transported” as part of the operation containing the request parameters or the response elements
- *Size* of the message in bytes
- *Message* which represents the entire SOAP message in a readable form

In terms of analyzing the *Transportation Security SensorNet* and its performance the *logging* module was engaged in all services. Quantitive results obtained using the logging capability can be found in [2], [3], [4]; a tool to visualize and animate the timing of the messages is described in [4].

c) *Addressing*: An implementation of the *WS-Addressing* specification as described in [36], [37] comes as part of the *addressing* module in the Axis2 core. It fully supports all components of the standard and its *ReplyTo* and *RelatesTo* fields are used among other things to allow for *asynchronous* communication (see IV-A6) in the TSSN.

d) *Savan*: The *Savan* module enables web services and clients in Axis2 to make use of various forms of subscription mechanisms as defined by the *WS-Eventing* specification [20].

e) *Rampart*: In order to provide security according to the *WS-Security* specification [38] for the TSSN the *Rampart* module was developed by Axis2. It makes extensive use of the *WS-SecurityPolicy* standard described by Lawrence et al. [39].

5) *Subscriptions*: Subscriptions are a fundamental part of the overall architecture of the TSSN. They are used by the *Alarm Processor* at the VNOC as well as in the MRN. These web services, that act as information publishers, utilize the *Savan* module to provide the operations defined in *WS-Eventing*.

6) *Synchronous and asynchronous communication*: By default Axis2 uses request-response in a *synchronous* manner. This means that the client has to wait and is therefore *blocking* until it receives the response from the service. In certain scenarios, for instance when the service needs a large amount of processing time, the client can experience timeouts. Furthermore, in the TSSN where the MRN is only intermittently connected to the VNOC, *synchronous* communication shows its limitations. A better option is to make the communication between services *asynchronous*. This resolves timeout issues and deals with connections that are only temporary. The following aspects need to be taken into consideration when using *asynchronous* communication:

a) *Client*: The client needs to make changes in regard to the how the request is sent out. Axis2 provides a low-level *non-blocking client API* and additional methods in the service stubs

that allow callbacks to be registered. These *AxisCallbacks* need to implement two methods, one that is being invoked whenever the response arrives and the other to define what happens in case of an error.

b) *Transport Level*: Depending on the transport protocol that is being used, Axis2 supports the following approaches.

- *One-way* uses one channel for the request and another one for the response such as the *Simple Mail Transfer Protocol* (SMTP)
- *Two-way* allows the same channel to be used for the request and the response, for example HTTP

For asynchronous communication to work the two-way approach was modified through the Axis2 *client API* which provides the option of using a *separate listener*. This tells the service that it is supposed to use a new channel for the response. In order to correlate request and response messages Axis2 makes use of the *WS-Addressing* specification, in particular the *RelatesTo* field.

c) *Service*: The final piece of asynchronous communication is to make the service processing asynchronous as well. This is done by specifying *asynchronous message receivers* in the services configuration in addition to the *synchronous* ones. Axis2 then uses the *ReplyTo* field of the *WS-Addressing* header in the client as a sign to send an immediate *acknowledge* of the request back to it. Furthermore it processes the request in a new thread and sends the response out when it is done, allowing the communication to be performed in asynchronous manner completely.

There exist various forms of transport protocols that are suitable for *asynchronous* communication. Axis2 by default supports HTTP, SMTP, JMS and TCP as transports but other transports can easily be defined and plugged in. The *Java Message Service* (JMS), for instance, makes use of *queues* which allow clients and services to store on them and retrieve messages in a flexible manner. This is essential for satellite communication which and discussed in detail in [3].

B. TSSN Common Namespace

Elements are often shared among a variety of services. Since defining the same element over and over again is neither a scalable nor maintainable approach, it makes sense to specify a common namespace for them and let the web services that want to use them, include them. In the TSSN these shared elements are part of the *TSSN Common* namespace.

C. Mobile Rail Network

The MRN is a collection of services that is located on a train or in a rail yard. Its services provide the abilities to manage sensors, monitor them and propagate sensor alerts to the VNOC. This section describes them in detail.

1) *Sensor Node*: The *Sensor Node* contains the actual sensor monitoring and management application and its components are shown in Figure 5. It provides several abstraction layers that allow various forms of sensors to be used. The current implementation makes use of cargo cable seals from Hi-G-Tek (HGT) [35]; these are considered one type of sensor

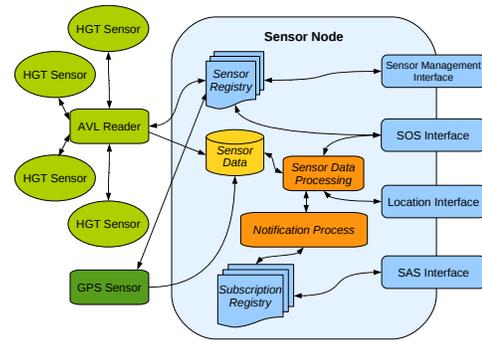


Fig. 5. Mobile Rail Network Sensor Node

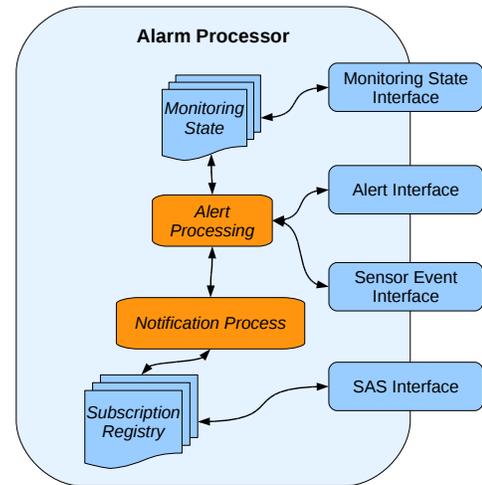


Fig. 6. Mobile Rail Network Alarm Processor

in the TSSN. Interaction with these sensors is performed using a (HGT) Automatic Vehicle Location (AVL) reader. The *Sensor Node* implements the functionality that allows higher level management of the sensors, e.g., here a collection of intelligent cargo seals connected to a series of containers, and the data that they provide through the use of a *sensor registry*, the *sensor data* storage and *sensor data processing*. Attaching a GPS sensor to the *Sensor Node* allows sensor events to be tagged with the specific location that they appeared at. The core functionality of the *Sensor Observation Service* that allows the service to offer its capabilities and observations is implemented. Furthermore, a *subscription registry* is available for alert notifications.

2) *Alarm Processor*: The *Alarm Processor* on the MRN performs an initial filtering of sensor events generated by the *Sensor Node*. It subscribes to of all events of the *Sensor Node*, providing interfaces for generic sensor events as well as sensor alerts. Alerts reported to the *Alarm Processor* include potential alarms that the *Sensor Node* reports, GPS acquisitions and losses, and status messages of the monitoring application such as when it is started and stopped. In case the data is not as complex as an alert, the *event* element provides a simple structure with a timestamp and a data field.

The *Alarm Processor* handles alerts and events that it

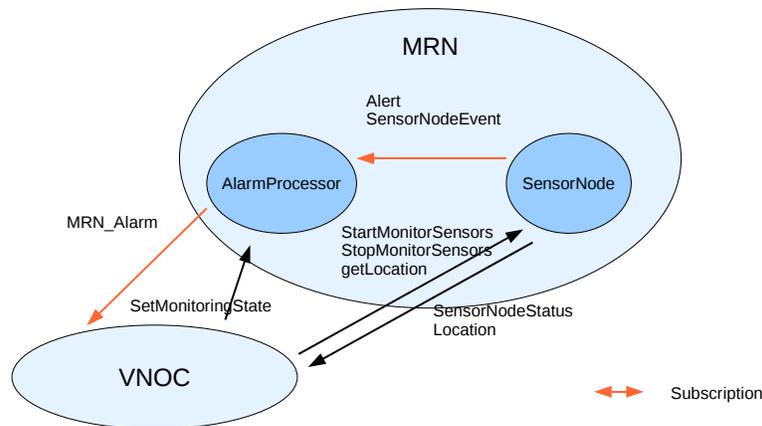


Fig. 4. Mobile Rail Network message overview

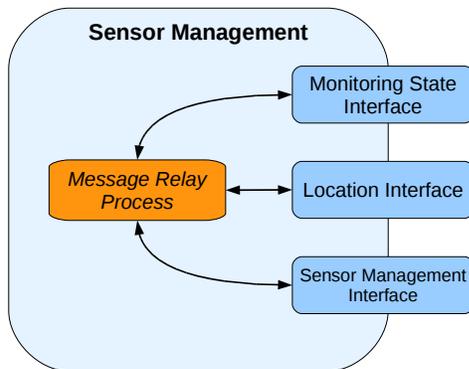


Fig. 8. Virtual Network Operation Center Sensor Management

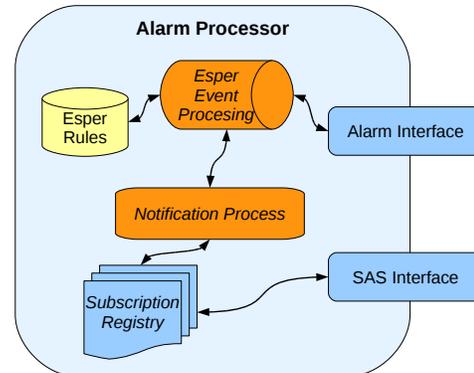


Fig. 9. Virtual Network Operation Center Alarm Processor

receives from the *Sensor Node* and classifies them into either *information* or *security* alarms depending on its current monitoring state. It is also responsible for deciding whether or not to forward the alarm to the VNOc for further processing and possible transmission to the decision maker.

D. Virtual Network Operation Center

The VNOc as shown in Figure 7 represents the management facility of the TSSN and consists of services that receive and process alerts received from MRN. It works with the TDE to associate shipment and trade information with a particular alert. Furthermore, the *Alarm Reporting* service provides clients with the ability to be notified upon specific events. The processes that are involved in performing these tasks are the topic of this section.

1) *Sensor Management*: The *Sensor Management* service (Figure 8) is responsible for controlling sensors and alarm reporting. It provides methods for starting and stopping sensor monitoring. Additionally the monitoring state which defines how alerts are interpreted and processed can be specified. The *Sensor Management* service essentially relays these “control” messages to the according MRN. Another functionality that is provided is the ability to query for a specific MRN’s location.

The implementation details of the interfaces that it provides to clients are described in the following.

The *Sensor Management* service allows the control of *Sensor Nodes* and their monitoring state. Additionally, it is able to retrieve the location of *Sensor Nodes*.

2) *Alarm Processor*: In contrast to the “basic” processing that is performed by the *Alarm Processor* at the MRN, the *Alarm Processor* as shown in Figure 9 at the VNOc has more resources such as the associated shipment and trade information available which is provided by the TDE and can therefore process alarms in a more complex way. This advanced filtering and processing is done using a complex event processing system called *Esper* developed by Bernhardt et al. [40].

Esper works on the basis of *sliding windows* in which events that are close together on the time axis are analyzed and correlated. It also supports using historical data from a variety of sources. An efficient query and filtering language called *Event Processing Language* allows for the most complex scenarios to be implemented. In the TSSN it is used for instance to filter out alarms for which shipment information could not be retrieved from the TDE and mark them as *security* notifications.

The *MRN_Alarm* operation is used as a notification interface

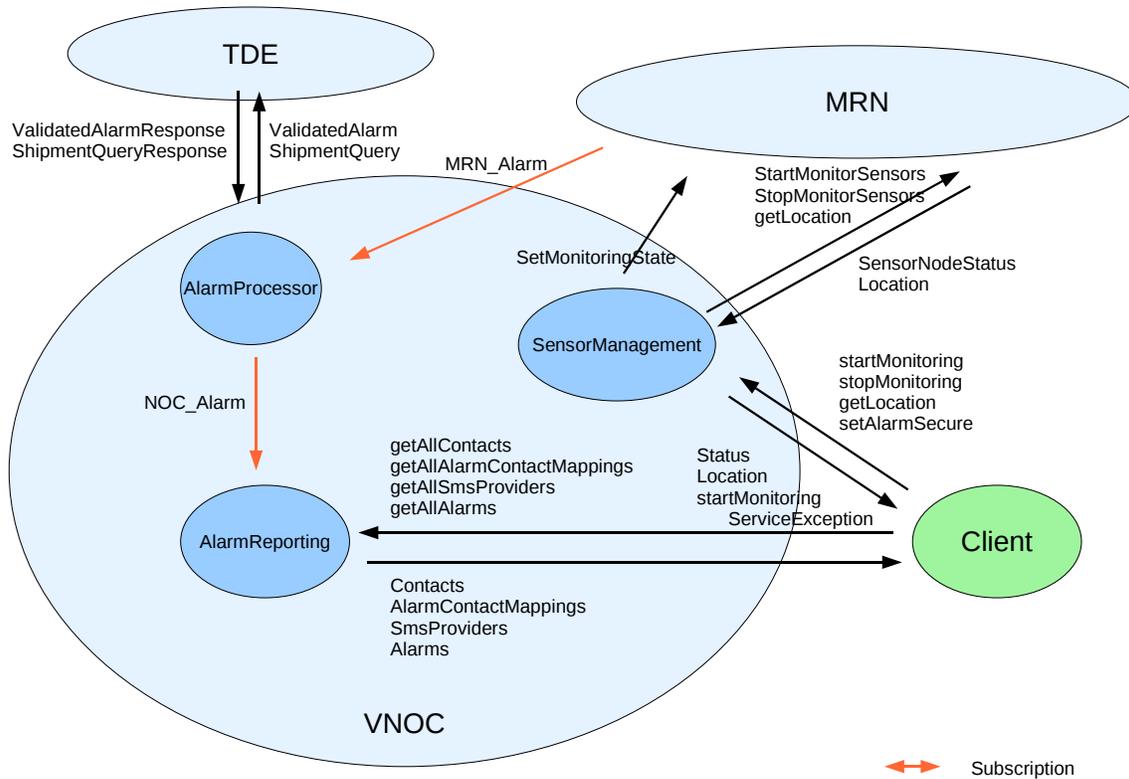


Fig. 7. Virtual Network Operation Center message overview

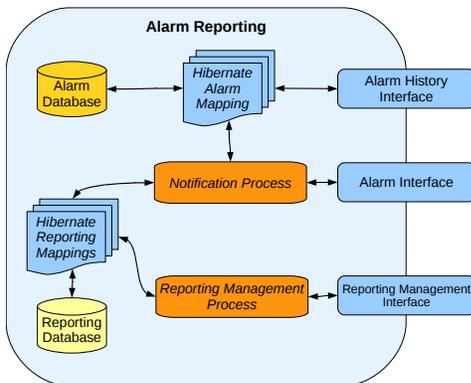


Fig. 10. Virtual Network Operation Center Alarm Reporting

for alarms from the *Alarm Processor* on the MRN. The *Alarm Processor* service subscribes to alarms from its counterpart on the MRN. Upon receiving an alarm, shipment data is retrieved from the TDE and attached to the original alarm. *Esper* then processes the alarm and passes it on to the *Alarm Reporting* service.

The *Alarm Processor* at the VNOc primarily provides functionality for the MRN to deliver alert notifications. It uses *Esper* to perform complex event processing, taking into consideration alert data and information from the TDE, and to forward alarms to the *Alarm Reporting* service.

3) *Alarm Reporting*: The *Alarm Reporting* service (Figure 10) deals with the following two aspects. First, it stores alarms long term to allow for in-depth reporting and analysis. Second, clients that want to be notified of particular alarms can register with the *Alarm Reporting* service. Whenever alarms occur notifications are sent out to the registered clients via email and/or SMS accordingly.

For long term data storage and to maintain a registry of the client notifications the *Alarm Reporting* service makes use of the *MySQL* database. In order to remain flexible and provide an abstraction layer to the core database functionality a tool called *Hibernate* [33] was utilized. An excellent introduction to the *object-relational* mapping is provided by Bauer et al. [41]. The main advantage is that objects referenced in code can easily be *persisted* into a relational database and vice versa. The only thing that needs to be defined is the *mapping*. Once that has been defined *Hibernate* takes care of the rest.

Since the objects that are being stored in the database are defined using XML schemas and then automatically compiled into Java objects during the build process, it makes sense to specify the mappings in XML as well. This is done in the TSSN. Another approach that is supported by *Hibernate* is using *annotations* within the Java objects themselves. This is not possible because of the aforementioned build process as the objects would have to be reannotated at every build.

The registry that is used for notifications contains *alarm contact mappings* that specify what kind of alarms a specific contact wants to be notified of. In case the contact wants to

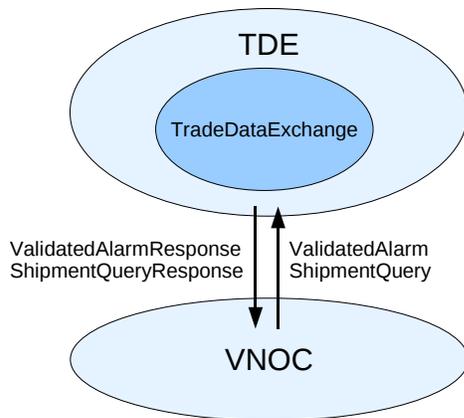


Fig. 11. Trade Data Exchange message overview

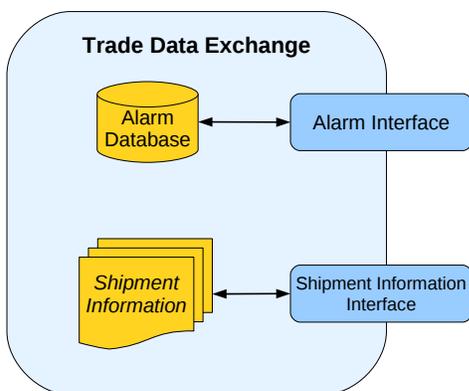


Fig. 12. Trade Data Exchange Service

receive SMS notifications, a SMS provider has to be specified as well.

The *Alarm Reporting* service receives alarm notifications from the *Alarm Processor* at the VNOC. It provides a notification interface primarily for the subscription of alarms from the *Alarm Processor*. The *Alarm Reporting* service subscribes to alarms and provides this operation for its notifications. An alarm here is a combination of the *tssn:MRN_AlarmBean* and shipment and trade information received from the TDE.

E. Trade Data Exchange

The *Trade Data Exchange* [42], as shown in Figure 11, in a sense represents a shipment and other trade data information provider. It aims to be a collection of heterogeneous systems that stores and manages the business aspects of a transport of goods. This is due to the fact that there is a variety of different systems implemented by the parties that participate in the transport chain (see II-A and II-C). Some provide route information while others manage contracts and shipment data. For the current implementation of the TSSN this “collection” of information and management services is combined into a single service, the TDE service.

The TDE service (Figure 12) interacts with the *Alarm Processor* at the VNOC. Upon request it provides shipment

and trade information for a specified alarm. It also provides functionality that can be used for long term alarm storage, although in its current implementation fairly limited. Since the service was designed externally, the elements used are not compatible to the TSSN common elements or any of the other services.

F. Open Geospatial Consortium Specifications

As described before, the amount of work that is required to fully implement OGC specifications such as the *Sensor Observation Service* and the *Sensor Alert Service* is immense. The focus of the first stage of the implementation of the TSSN is on the sensor management and alarm notification capabilities. However, at the MRN the *Sensor Node* provides an implementation for the *Sensor Observation Service* as defined by the OGC. Furthermore, services in the TSSN that utilize subscriptions, in particular the *Alarm Processor*, are able to receive *subscribe* requests and publish *alerts* in a manner that is similar to the *Sensor Alert Service*. The difference to the proposed SAS specification is that the services that subscribe are already aware of the *capabilities*, *sensor* types and *alert* types. Therefore the operations that allow the retrieval of this information need to be implemented in order to be fully compliant.

V. RESULTS

Several experiments were performed at various development stages of the TSSN. First, lab tests were conducted in order to ensure the functionality of the individual web services and their interactions.

Then, as described in [2], truck trials were completed to test basic interaction of the implemented web services and feasibility of hardware components and sensors in a mobile environment. The message exchanges between web services were correct and the system was able to recover from dropped communication links and lost GPS fixes. In addition, it was found that the read range of the sensors used is about 400 meters.

A short haul rail trial was conducted after the successful completion of truck tests. Results of the short haul rail trial are found in [2]. One of the goals was to determine the performance of the TSSN when detecting events on intermodal containers in a rail environment. Furthermore SMS message and email notification of events was investigated and data collected that could be used in the modeling of system trade-offs and communication models. The system performed well: the time it took from detecting an event to generating an alert was about 2 seconds and the average delivery time of the alert was about 12 seconds. This is well within the bounds of the requirements of the transportation industry for efficient tracking and monitoring of cargo. Note that for the short haul trial a GSM communication link was used that proved to be stable and reliable. This allowed the web services to interact synchronously with each other.

Enhancements made to the TSSN to work well in low bandwidth mobile bandwidth limited and intermittently connected monitoring environments are described in [3]. In particular, the

communication link between the MRN and the VNOC was changed to a dial-up satellite connection and the web services were adapted to utilize a distributed queuing approach and hence communicate asynchronously. The viability of these adjustments was tested in a long haul rail trial in Mexico. Again the TSSN worked well and was able to transmit messages in about 12 seconds whenever connectivity was established. In case the satellite link was down and needed to be established it took about 10 minutes on average to deliver messages from the MRN to the VNOC. However the average case of about 7 minutes per message transmission through the system is found to be in range of mobile monitoring environments.

VI. CONCLUSION

The implementation of the *Transportation Security SensorNet* using a *Service Oriented Architecture* works. Testing has been completed in a lab environment as well as in the real world and TSSN was evaluated in [2], [3], [4]. The complete system provides a *web services* based sensor management and alarm notification infrastructure that is built using open standards and specifications. Particular functionality within the system has been implemented in *web services* that provide interfaces according to their respective *web service* specifications.

Using standards from the *Open Geospatial Consortium* allows the integration of the system into *Geographic Information Systems*. Although not all the interfaces are fully implemented as of summer 2009, the basic *Sensor Observation Service* and *Sensor Alert Service* are. Other OGC specifications can be integrated a lot easier now because enhancements to the Axis2 schema compiler have been made by the author (see IV-A1b).

WS-Eventing plays an important role in the *Transportation Security SensorNet* as it is essential for the alarm notification chain. The specification that is used by all the clients and services is *WS-Addressing*. Note that HTTP, which represents the underlying *transport layer* of most the *web services*, already provides an addressing scheme. This however, is not as useful as it seems because web services may change their *transport layer* and messages sometimes require complex routing. The reasoning behind this and other things have been explained in detail.

Overall the TSSN provides a *Service Oriented Architecture for Monitoring Cargo in Motion Along Trusted Corridors*. This *web services* based approach allows for platform and programming language independence and offers compatibility and interoperability. The integration of SOA, OGC specifications and sensor networks is complex and difficult. As described in III-H, most systems and research focuses either on the combination of SOA and OGC specifications or on OGC standards and sensor networks. However, the TSSN shows that all three areas can be combined and that this combination provides capabilities to the transportation and other industries that have not existed before. In particular, web services in a mobile sensor network environment have always been seen as slow and producing a lot of overhead. The TSSN, as shown by the results in [2], [3] demonstrates that with proper architecture and design the performance requirements of the targeted scenario can be satisfied.

Furthermore, the *Transportation Security SensorNet* and its *Service Oriented Architecture* allow sensor networks to be utilized in a standardized and open way through web services. Sensor networks and their particular communication models led to the implementation of asynchronous message transports in SOA and are supported by the TSSN.

VII. FUTURE WORK

After evaluating the current implementation, several points of improvement were identified.

a) *Security*: The current system only provides entry points for the *WS-Security* in terms of the *Rampart* module. There are several issues in the current implementation of the module, especially with regard to attaching policies to *web services* and clients. This is discussed in [5]. Further development is underway to implement *WS-Security*. In between the *Virtual Network Operation Center* and the *Mobile Rail Network* communication is secured by establishing a *Virtual Private Network* (VPN). However, this is not practical using a satellite link because of performance reasons. Sensors management is done at the *Sensor Node* but as of now there is no support for the secure handover to other *Sensor Nodes*. The remote management systems need to be improved in this area.

b) *Service Discovery*: Due to several problems in the specific implementation of the UDDI that was used, for the trials most of the services were made aware of the other services through the means of configuration instead of service discovery. Since using a UDDI provides far better scalability, it is an essential piece of future versions of the TSSN.

c) *Multiple service clouds*: During the trials all services were unique which in an operational system this is not the case. There are issues that need to be explored in dealing with multiple versions not only of single *web services* but multiple VNOC's and MRN's. This is especially important when it comes to managing policies and subscriptions properly.

ACKNOWLEDGMENT

This work was supported in part by Oak Ridge National Laboratory (ORNL) Award Number 4000043403. This material is also partially based upon work supported while V. S. Frost was serving at the National Science Foundation.

The authors wish to acknowledge Kansas City Southern Railway for their participation in the rail trials. We would also like to acknowledge the support of EDS, an HP company, and Kansas City SmartPort, Inc. our partners on this project.

REFERENCES

- [1] M. Wolfe, "In this case, bad news is good news," *Journal of Commerce*, July 2004, www.ismasecurity.com/ewcommon/tools/download.aspx?docId=175.
- [2] D. Fokum, V. Frost, D. DePardo, M. Kuehnhausen, A. Oguna, L. Searl, E. Komp, M. Zeets, D. Deavours, J. Evans, and G. Minden, "Experiences from a transportation security sensor network field trial," in *GLOBE-COM Workshops, 2009 IEEE*, 30 2009-Dec. 4 2009, pp. 1–6.
- [3] M. Kuehnhausen and V. S. Frost, "Application of the Java Message Service in Mobile Monitoring Environments," Information Telecommunication and Technology Center, University of Kansas, Lawrence, KS, Tech. Rep. ITTC-FY2010-TR-41420-18, February 2010.

- [4] —, “Framework for Analyzing SOAP Messages in Web Service Environments,” Information Telecommunication and Technology Center, University of Kansas, Lawrence, KS, Tech. Rep. ITTC-FY2010-TR-41420-20, March 2010.
- [5] E. Komp, V. S. Frost, and M. Kuehnhausen, “Implementing Web Services: Conflicts Between Security Features and Publish/Subscribe Communication Protocols,” Information Telecommunication and Technology Center, University of Kansas, Lawrence, KS, Tech. Rep. ITTC-FY2010-TR-41420-19, February 2010.
- [6] D. Group, “The value of standards,” Delphi Group, Ten Post Office Square, Boston, MA 02109, Survey, Jun. 2003, www.ec-gis.org/sdi/ws/costbenefit2006/reference/20030728-standards.pdf.
- [7] M. Reichardt, “The Havoc of Non-Interoperability,” OGC, OGC White Paper, Dec. 2004, http://portal.opengeospatial.org/files/?artifact_id=5097.
- [8] L. McKee, “The Importance of Going ‘Open,’” OGC, OGC White Paper, Jul. 2005, http://portal.opengeospatial.org/files/?artifact_id=6211.
- [9] M. Imren, “10 ways to reduce the cost and risk of global trade management,” *Journal of Commerce*, March 2009, <http://www.joc.com/node/410216>.
- [10] L. F. Cabrera, C. Kurt, and D. Box, “An Introduction to the Web Services Architecture and Its Specifications,” Microsoft, Microsoft Technical Article, Oct. 2004, <http://msdn.microsoft.com/en-us/library/ms996441.aspx>.
- [11] D. Nickul, L. Reitman, J. Ward, and J. Wilber, “Service Oriented Architecture (SOA) and Specialized Messaging Patterns,” Adobe, Adobe Article, Dec. 2007, www.adobe.com/enterprise/pdfs/Services_Oriented_Architecture_from_Adobe.pdf.
- [12] M. Botts, G. Percivall, C. Reed, and J. Davidson, “OGC Sensor Web Enablement: Overview And High Level Architecture,” OGC, OGC White Paper, Dec. 2007, http://portal.opengeospatial.org/files/?artifact_id=25562.
- [13] X. Chu, T. Kobialka, and R. Buyya, “Open sensor web architecture: Core services,” in *In Proceedings of the 4th International Conference on Intelligent Sensing and Information Processing*. Press, 2006, pp. 1–4244, <http://www.gridbus.org/papers/ICISIP2006-SensorWeb.pdf>.
- [14] X. Chu, “Open sensor web architecture: Core services,” Master’s thesis, University of Melbourne, Australia, 2005, <http://www.gridbus.org/reports/OSWA-core%20services.pdf>.
- [15] D. Fitzpatrick, D. Dreyfus, B. A. Hamilton, M. Onder, and J. Sedor, “The electronic freight management initiative,” U.S. Department of Transportation, Federal Highway Administration, Tech. Rep. FHWA-HOP-06-085, April 2006.
- [16] K. Troup, D. Newton, M. Wolfe, and R. Schaefer, “Columbus electronic freight management evaluation - achieving business benefits with efm technologies,” Science Applications International Corporation (SAIC), Tech. Rep., March 2009.
- [17] I. Foster, C. Kesselman, J. M. Nick, and S. Tuecke, “The physiology of the grid: An open grid services architecture for distributed systems integration,” in *Open Grid Service Infrastructure WG, Global Grid Forum*, Jun. 2002, <http://www.globus.org/alliance/publications/papers/ogsa.pdf>.
- [18] T. Bellwood, L. Clement, D. Ehnebuske, A. Hately, M. Hondo, Y. L. Husband, K. Januszewski, S. Lee, B. McKee, J. Munter, and C. von Riegen, “UDDI Version 3.0,” OASIS, OASIS Specification, Jul. 2002, <http://uddi.org/pubs/uddi-v3.00-published-20020719.htm>.
- [19] T. Bellwood, “Rocket ahead with UDDI V3,” IBM, IBM Article, Nov. 2002, <http://www.ibm.com/developerworks/webservices/library/ws-uddiv3/>.
- [20] D. Box, L. F. Cabrera, C. Critchley, F. Curbera, D. Ferguson, S. Graham, D. Hull, G. Kakivaya, A. Lewis, B. Lovering, P. Niblett, D. Orchard, S. Samdarshi, J. Schlimmer, I. Sedukhin, J. Shewchuk, S. Weerawarana, and D. Wortendyke, “Web services eventing (ws-eventing),” W3C, W3C Member Submission, Mar. 2006, <http://www.w3.org/Submission/2006/SUBM-WS-Eventing-20060315/>.
- [21] A. Friis-Christensen, N. Ostländer, M. Lutz, and L. Bernard, “Designing service architectures for distributed geoprocessing: Challenges and future directions,” *Transactions in GIS*, vol. 11, no. 6, pp. p799 – 818, 20071201. [Online]. Available: <http://search.ebscohost.com.www2.lib.ku.edu:2048/login.aspx?direct=true&db=aph&AN=28048261&site=ehost-live>
- [22] C. Kiehle, K. Greve, and C. Heier, “Requirements for next generation spatial data infrastructures-standardized web based geoprocessing and web service orchestration,” *Transactions in GIS*, vol. 11, no. 6, pp. p819 – 834, 20071201. [Online]. Available: <http://search.ebscohost.com.www2.lib.ku.edu:2048/login.aspx?direct=true&db=aph&AN=28048260&site=ehost-live>
- [23] C. M. MacKenzie, K. Laskey, F. McCabe, P. F. Brown, R. Metz, and B. A. Hamilton, “Reference Model for Service Oriented Architecture 1.0,” OASIS, OASIS Standard, Oct. 2006, <http://docs.oasis-open.org/soa-rm/v1.0/>.
- [24] H. Haas, D. Booth, E. Newcomer, M. Champion, D. Orchard, C. Ferris, and F. McCabe, “Web services architecture,” W3C, W3C Note, Feb. 2004, <http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/>.
- [25] M. Kalin, *Java Web Services: Up and Running*. O’Reilly, February 2009.
- [26] L. S. Searl, “Service Oriented Architecture for Sensor Networks Based on the Ambient Computing Environment,” ITTC, ITTC Technical Report, Feb. 2008, www.ittc.ku.edu/sensornet/trusted_cooridors/papers/41420-07.pdf.
- [27] E. Chinthaka, “Web services and Axis2 architecture,” IBM, IBM Article, Nov. 2006, <https://www.ibm.com/developerworks/webservices/library/ws-apacheaxis2/>.
- [28] A. S. Foundation, “XMLBeans,” Jul. 2008. [Online]. Available: <http://xmlbeans.apache.org/>
- [29] J. Fialli and S. Vajjhala, “Java architecture for xml binding (jaxb) 2.0,” Java Specification Request (JSR) 222, October 2005.
- [30] D. Sosnoski, “JiXB,” Mar. 2009. [Online]. Available: <http://jibx.sourceforge.net/>
- [31] Y. Lafon and N. Mitra, “SOAP version 1.2 part 0: Primer (second edition),” W3C, W3C Recommendation, Apr. 2007, <http://www.w3.org/TR/2007/REC-soap12-part0-20070427/>.
- [32] D. Booth and C. K. Liu, “Web services description language (WSDL) version 2.0 part 0: Primer,” W3C, W3C Recommendation, Jun. 2007, <http://www.w3.org/TR/2007/REC-wsdl20-primer-20070626>.
- [33] R. Hat, “Hibernate Reference Documentation 3.3.1,” Tech. Rep., Sep. 2008, http://www.hibernate.org/hib_docs/v3/reference/en-US/pdf/hibernate_reference.pdf.
- [34] EsperTech, “Esper - Event Stream and Complex Event Processing for Java.” [Online]. Available: <http://www.espertech.com/>
- [35] Hi-G-Tek. [Online]. Available: <http://www.higtek.com/>
- [36] M. Gudgin, M. Hadley, and T. Rogers, “Web services addressing 1.0 - core,” W3C, W3C Recommendation, May 2006, <http://www.w3.org/TR/2006/REC-ws-addr-core-20060509>.
- [37] M. Gudgin, M. Gudgin, M. Hadley, T. Rogers, T. Rogers, and M. Hadley, “Web services addressing 1.0 - SOAP binding,” W3C, W3C Recommendation, May 2006, <http://www.w3.org/TR/2006/REC-ws-addr-soap-20060509>.
- [38] K. Lawrence, C. Kaler, A. Nadalin, R. Monzillo, and P. Hallam-Baker, “Web Services Security: SOAP Message Security 1.1 (WS-Security 2004),” OASIS, OASIS Standard, Feb. 2006, <http://docs.oasis-open.org/wss/v1.1/wss-v1.1-spec-os-SOAPMessageSecurity.pdf>.
- [39] K. Lawrence, C. Kaler, A. Nadalin, M. Goodner, M. Gudgin, A. Barbir, and H. Granqvist, “WS-SecurityPolicy 1.2,” OASIS, OASIS Standard, Jul. 2007, <http://docs.oasis-open.org/ws-sx/ws-securitypolicy/v1.2/ws-securitypolicy.pdf>.
- [40] T. Bernhardt and A. Vasseur, “Event-driven application servers,” 2007. [Online]. Available: http://dist.codehaus.org/esper/JavaOne_TS-1911_May_11_2007.pdf
- [41] C. Bauer and G. King, *Hibernate in Action*. Manning, 2005.
- [42] K. SmartPort, “Trade Data Exchange - Nothing short of a logistics revolution,” *Journal of Commerce*, November 2008. [Online]. Available: <http://www.joc-digital.com/joc/20081110/?pg=29>

The University of Kansas



Technical Report

**A Dual-Resonant Microstrip-Based
UHF RFID "Cargo" Tag**

Supretha Aroor and Daniel D. Deavours

ITTC-FY2010-TR-41420-23

March 2008

Project Sponsor:
Oak Ridge National Laboratory
Award Number 4000043403

Copyright © 2008:
The University of Kansas
2335 Irving Hill Road, Lawrence, KS 66045-7559
All rights reserved.

A Dual-Resonant Microstrip-Based UHF RFID “Cargo” Tag

Supreetha Aroor and Daniel D. Deavours Information and Telecommunications Technology

Center

University of Kansas, Lawrence, KS

Email: {saroor,deavours}@ittc.ku.edu

Abstract

We present a novel passive UHF RFID “cargo tag” capable of operating in two different geographic regions such as Europe and North America. The tag utilizes a dual-resonant, dual-polarized, microstrip antenna. The resulting tag operates efficiently over ETSI and FCC frequency ranges and achieves excellent efficiency. The tag antenna eliminates any cross-layer structures such as a via and may be manufactured efficiently using traditional, low-cost “inlay” technology. We estimate the free-space read distance to be between 9.3 and 13.1 meters (30 and 43 feet).

Index Terms

Antennas, RFID, Antenna feeds, Microstrip antennas, Multifrequency antennas, Impedance matching.

This work was supported by the Information and Telecommunications Technology Center at the University of Kansas, the Office of Naval Research through Award Number N00014-07-1-1042, Oak Ridge National Laboratory via Award Number 4000043403, and the KU Transportation Research Institute.

CONTENTS

I	Introduction	3
II	Background and Related Work	3
III	Antenna Design	5
IV	Measurements	7
V	Conclusion	9
	References	10

LIST OF FIGURES

1	Dual-resonant, dual-polarized planar microstrip-based RFID tag.	6
2	Simulated currents on antenna.	6
3	Power wave Smith chart of measured antenna impedance (normalized to $13 - j65 \Omega$). The 90% and 50% power transfer efficiency circles are also plotted.	7
4	Radiation pattern.	9

I. INTRODUCTION

Inexpensive, passive UHF RFID antennas are typically constructed using stripline dipole antennas [1]. The antennas are often electrically short in order to fit on a 4-inch label and narrow in order to minimize cost. Dipole antennas experience degraded performance when they are placed on or close to metal objects [2], [3], [4], and thus are not well suited for tracking large metal assets.

One of the practical advantages of modern, efficient UHF RFID ICs is the ability to identify tags at long distances. This is particularly useful for identifying metal containers, such as cargo containers. Such assets are ideal applications of UHF RFID for many reasons: they tend to have a long useful life, allowing the tag cost to be amortized over many uses; the contents change frequently, so associations with a unique identifier is useful; they frequently cross organizational boundaries, making a common mode of identification useful; and they are physically large, requiring identification at distance. Further, cargo containers are commonly shipped between large geographic regions, and different regions have different frequencies allocated for RFID. Worldwide, the UHF RFID frequency spectrum ranges roughly between 860–960 MHz range. Practical considerations require a low profile and moderate form factor, and cost is often a primary driver in such systems.

In this paper, we present a nearly four-inch square microstrip antenna placed over a six-inch square dielectric and ground plane adopted for low-cost RFID tagging of large, metal objects such as cargo containers. The antenna is dual-resonant so as to operate in two of the three frequency bands used for UHF RFID worldwide. Finally, due to a large antenna gain from the microstrip antenna, e.g., 5 dB larger than a dipole, we were able to validate the tag performance informally by reading the tag over 60 feet (18.3 meters) away in an outdoor environment using a commodity RFID reader (likely utilizing a ground reflection for 3 dB performance improvement). We estimate the free-space read distance to be between 9.3 and 13.1 meters.

II. BACKGROUND AND RELATED WORK

Microstrip “patch” antennas are a well-known class of antennas (e.g., [5]). A microstrip antenna consists of: the primary radiating element (“antenna”), a dielectric substrate, a ground plane, and a feeding element. Common feeds include a probe feed from a coaxial cable or edge feed with a microstrip transmission line, although proximity feeds and coupling through a slot in the ground plane are also possible.

Traditional approaches to microstrip-based RFID tags take these familiar structures and adapt them to the particularities of RFID. Unfortunately, the most common feed structures are based on some sort of unbalanced transmission line, whether it be a coaxial cable with a probe feed, or a microstrip transmission

line with an edge feed. The need to establish an electrical reference (“ground”) complicates the structure, usually requiring a via to the ground plane (e.g., [6]).

The via significantly complicates the manufacturing process. The UHF RFID industry commonly uses high-speed, web-based processing to manufacture “inlays.” An *inlay* is a printed or etched antenna on a flexible substrate with a RFID chip attached, which can be manufactured with a web-based process in high volumes and at low cost. Another challenge with traditional microstrip antennas is a narrow bandwidth and single resonant frequency. Unfortunately, different regions utilize different frequency ranges within the UHF spectrum for RFID use. For example, North America uses 902–928 MHz, Europe has allocated 865–868 MHz, and Japan allocated 952–954 MHz. Most other countries of the world utilize some subset of 865–868 or 902–928 MHz. It becomes a significant challenge to make a tag operate well over the entire 902–928 MHz frequency range and retain a low profile, small form factor, and high performance. Operating over two bands, e.g., cargo containers shipping supplies between North America and Europe, becomes especially challenging.

By feeding a rectangular microstrip antenna along the diagonal (with a probe feed), one can excite both the TM_{01} and TM_{10} modes [7], where the two modes have orthogonal polarization, if the two modes are sufficiently separated. The antenna dimensions can be shortened and the resonant frequencies controlled by cutting a cross-shaped slot into the antenna [8].

Commonly, antennas are designed to match to a substantially real load, e.g., a 50 or 75 Ohm transmission line, in order to minimize standing waves along the transmission line. With RFID antennas, the load is both in close proximity to the antenna and substantially reactive. Thus, the objective to optimize RFID tag performance is not to minimize the voltage reflection coefficient, but rather to maximize the power transferred to the IC (load). The remainder of this section follows from [9], [10].

Let Z_a and Z_c be the antenna and chip impedance respectively. Maximum power transfer is achieved when $Z_a = Z_c^*$, in which half the power is transferred to the load and half is scattered. The *power transfer efficiency* is the ratio of the actual power transferred to the maximum possible power transferred (a conjugate match), which is given by

$$\tau = \frac{4R_a R_c}{|Z_a + Z_c|^2}.$$

Similarly, a *power wave reflection* can be defined as

$$s = \frac{Z_c - Z_a^*}{Z_a + Z_c},$$

so that $\tau + |s|^2 = 1$. Here, s plays a similar role to S_{11} in the traditional antenna matching problem.

Let

$$\hat{z}_a = r + jx = \frac{R_a}{R_c} + j \frac{X_a + X_c}{R_c}.$$

It can be shown that $s = \frac{\hat{z}_a - 1}{\hat{z}_a + 1}$, i.e., s is a Smith chart transformation. The antenna (source) impedance can be plotted on a Smith chart and the distance from \hat{z}_a to the center of the Smith chart is $|s|$. We call the Smith chart normalized in this way a *power wave Smith chart*.

III. ANTENNA DESIGN

The inspiration for the antenna design comes from a combination of three factors. First, we draw from the dual-resonant rectangular antennas closely exciting the TM_{01} and TM_{10} modes and feeding in such a way to excite both modes [7]. Second, we draw from the use of microstrip transmission lines to build a completely planar microstrip antenna [11]. Third, we note the use of a cross-shaped slot in a microstrip antenna originally used to reduce the size of the antenna [8], but instead we use the cross as space to place our matching circuit and IC. The slots can also be used to resize the antenna as necessary to control the form factor.

Fig. 1 shows the antenna geometry, where the darkened area represents the metalized area. The antenna was designed to operate over a polypropylene substrate 5.08 mm thick and a dielectric constant of 2.28 and loss tangent estimated to be 0.001. The substrate and ground plane are 6 inches (15.2 cm) square. The initial design to use a simple cross resulted in a matching circuit with a reactance that was too large for our IC. To reduce the reactance, we shortened the transmission lines in the matching circuit by imposing a diamond-shape cut-out within the cross.

The cross segments have a length of 34 mm and width of 8 mm. The diamond is approximately 19.8 mm per side. The center of the feed lines are 15 mm from the center and 2 mm wide in order to achieve the desired input resistance, and the length of the feeds can be adjusted to achieve a reactance of approximately $j65 \Omega$ at resonance. The matching circuit was designed to obtain a perfect conjugate match to $13 - j65 \Omega$ at 867 MHz, and a slightly larger real resistance at 915 MHz to to achieve larger bandwidth.

When the first resonant mode is excited, one of the feeds acts as an inset microstrip feed, while the other feed lies primarily along the axis of symmetry and thus acts as a feed from an electrical reference, i.e., the virtual ground. When the second mode is excited, the role of the two feeds reverse. Fig. 2 illustrate the simulated currents on the antenna at the two resonant frequencies. One can see that the TM_{01} and TM_{10} modes are clearly excited with only a minor diagonal component excited at each frequency.

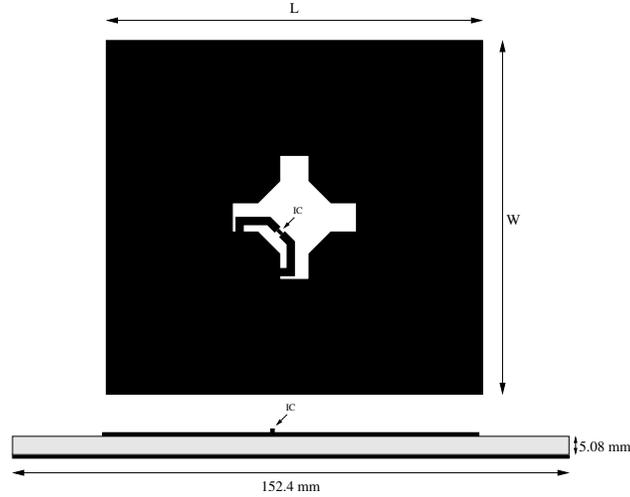


Fig. 1. Dual-resonant, dual-polarized planar microstrip-based RFID tag.

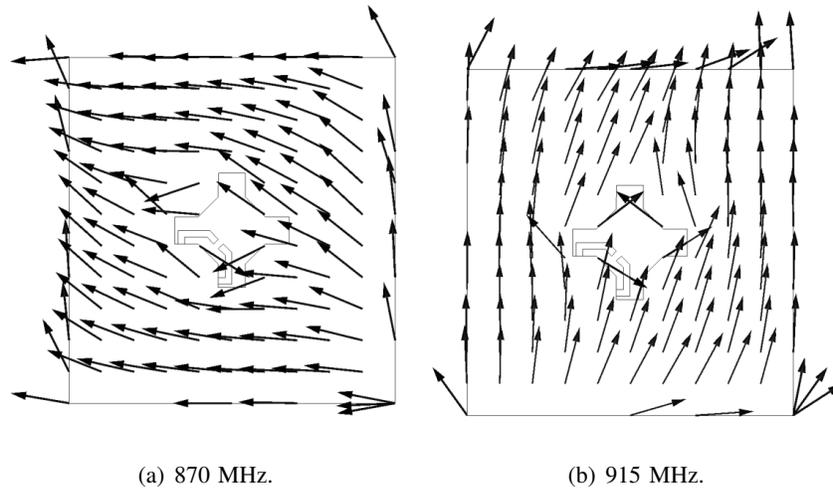


Fig. 2. Simulated currents on antenna.

The two resonant frequencies can be easily adjusted by adjusting L and W . Here, we present resonant frequencies of 867 and 915 MHz using $L = 103$ mm and $W = 97$ mm. To achieve resonant frequencies of 915 and 953 MHz, set $L = 97$ mm and $W = 92.5$ mm. For resonant frequencies at 867 and 953 MHz, set $L = 103$ mm and $W = 92.5$ mm. Matching to a larger or smaller resistance is achieved by placing the feed lines further apart or closer together, respectively. If the impedance is too small, the size of the cross may need to be expanded to facilitate a larger R_a . A smaller X_a can be achieved by shortening the feed lines, and a larger X_a by longer feed lines. In this way, the antenna can be adjusted to work

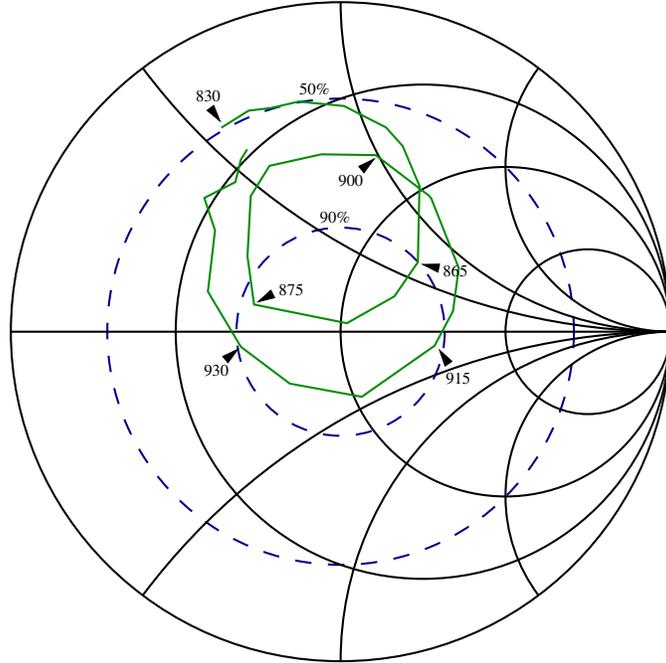


Fig. 3. Power wave Smith chart of measured antenna impedance (normalized to $13 - j65 \Omega$). The 90% and 50% power transfer efficiency circles are also plotted.

at any two of the three major frequencies of operation worldwide and with most commercially-available RFID ICs.

IV. MEASUREMENTS

As the antenna impedance is considerably reactive and balanced, it is difficult to use standard microwave measurement techniques. We constructed a measurement apparatus using a chip balun to measure the impedance using a network analyzer. The balun is mounted on a small PCB that converts a SMA signal to a differential signal to short pins, which are used to probe the IC connection pads. The apparatus yields only modest accuracy, but sufficient for our purposes here. The impedance measurements are taken over the range of 830 to 960 MHz. Note also that the impedance is near the edge of a 50 Ohm Smith chart, and thus the accuracy of the network analyzer is likely to be relatively poor. The power wave Smith chart (cf. Section II) using the normalized impedance $13 - j65 \Omega$ is plotted in Fig. 3.

Note that we are able to achieve an almost perfect conjugate match at 870 MHz. the 90% transfer efficiency spans between 864 and 875 MHz, and between 915 and 930 MHz. At 900 MHz, $\tau = 70\%$. By making the antenna slightly less inductive, the 1 dB bandwidth would span between 860 and 940

MHz, and the 3 dB bandwidth would cover from less than 830 to more than 960 MHz.

The simulated results of the antenna using a finite element analysis code [12] indicates a directivity of 6.2 and 6.1 dBi and radiating efficiency of -0.36 and -0.31 dB at 870 and 915 MHz, respectively. The measured impedance indicates $\tau \approx 0$ dB at 870 MHz and -0.4 dB at 915 MHz. Combining terms, we would expect a realized gain (the product of antenna gain and power transfer efficiency) of 5.9 and 5.4 dBi, respectively.

The free-space Friis equation can be modified for RFID transponders [13].

$$r = \frac{\lambda}{4\pi} \sqrt{\frac{P_t G_t G_r \tau \rho}{P_{th}}}$$

We following the convention of [13] that the subscript t represents the transmitter (reader) and the subscript r represents the receiver (tag); τ is the power transfer efficiency, ρ is the polarization mismatch, and P_{th} is the minimum (threshold) power to operate the IC. Commonly, readers use 6 dBi circularly polarized antennas and tag antennas are linearly polarized, leading to a polarization loss of 0.5. Using a published value of $P_{th} = -13$ dBm [14], the maximum read distance of the tag from a commodity reader with $P_t = 30$ dBm and $G_r = 5.4$ dBi is approximately 9.7 meters (32 feet). Utilizing a 3 dB ground reflection, one could expect 13.7 meters (45 feet).

We performed an informal read distance experiment outdoors in an open field. The tag was first affixed to an expanded polystyrene sheet and held approximately one meter above the ground and far from any other object. We used a commodity reader at maximum FCC power settings (36 dBm EIRP) with a bistatic circularly-polarized antenna. The tag was moved away from the reader until we found the maximum distance at which the tag was detectable (maximum detectable distance). We anticipate a 3 dB increase in received power from a ground reflection. We found the maximum detectable distance to be 18.3 meters (60 feet). This exceeds our estimated maximum detection distance by 2.5 dB. Next, we placed the antenna over a large ground plane (approximately 2 feet square) and observed the maximum detection increase to 20.5 meters (67 feet). The difference of about 1 dB change in directivity agrees with predictions based on simulation. Given those results, we would predict a free-space maximum detection distance of 12.9 and 14.5 meters (42 and 47 feet) without and with a large ground plane, respectively.

As a second method of validation, we placed the tag two meters from a different reader in a laboratory environment. The tag was placed on an expanded polystyrene block 2 meters from the reader. The reader frequency was fixed to 915 MHz, and the power was varied until we found the minimum power that was able to read the tag. Based on -13 dBm IC turn-on power, we would predict the minimum reader power would be 16.5 dBm. We measured 17.0 dBm. Given that result, we would predict a free-space read

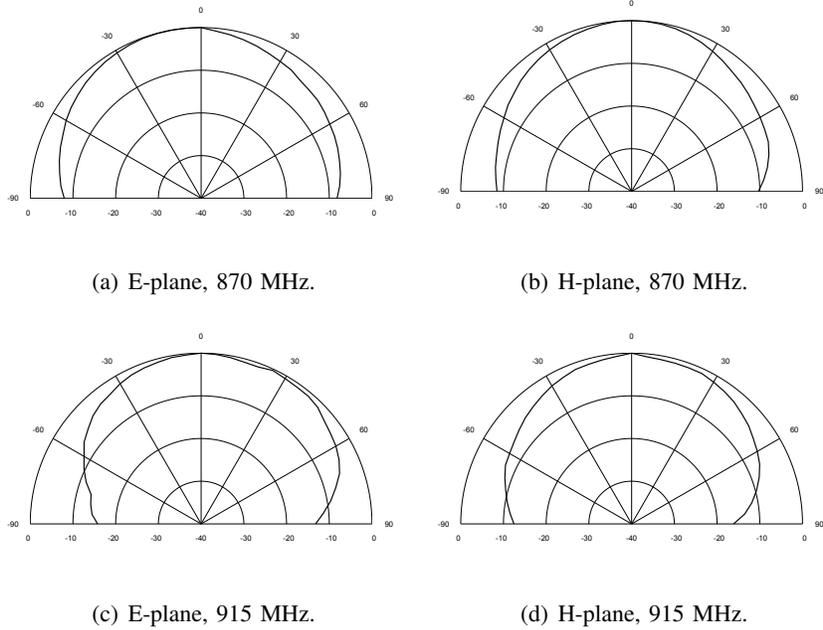


Fig. 4. Radiation pattern.

distance of 9.3 meters (32 feet). However, we have reason to believe that the system is not forward-link limited [4], and thus results in a conservative estimate. Regardless, we can conclude that the free-space read distance will be between 9.3 and 13.1 meters, and 10.4 and 14.7 meters on a large metal ground plane. Obviously, matched linearly polarized reader antennas will add 3 dB of link margin.

The measured radiation pattern of the tag in both the E- and H-plane at both 870 and 915 MHz are shown in Fig. 4. Note that since the antenna is dual-polarized, the planes reverse at the two resonant frequencies. The radiation pattern was measured with the provided (not infinite) ground plane, and show good agreement with simulated results.

V. CONCLUSION

In this paper, we present a high-performance, dual-resonant, dual-polarized microstrip antenna. The antenna is a completely planar structure (avoids any cross-layer connections) so that the antenna may be easily manufactured using a traditional inlay. The antenna is designed to perform efficiently within the 865–868 and 902–928 MHz frequency bands, but we show how to modify the design to operate at any two of the three frequency bands used worldwide. We show measured impedance results using the power wave Smith chart, showing excellent power transfer efficiency across the frequency bands of interest. Measured results show an estimated free-space detection distance of between 9.3 and 13.1 meters.

ACKNOWLEDGMENT

We are grateful for the helpful remarks from Kenneth R. Demarest and Daniel M. Dobkins.

REFERENCES

- [1] D. M. Dobkins, *The RF in RFID: Passive UHF RFID in Practice*. Burlington, MA: Newnes, 2007.
- [2] D. M. Dobkins and S. Weigand, "Environmental effects on RFID tag antennas," in *IEEE MTT-S International Microwave Symposium*, Long Beach, CA, Jun. 2005, pp. 4–7.
- [3] J. Griffin, G. Durgin, A. Haldi, and B. Kippelin, "Radio link budgets for 915 MHz RFID antennas placed on various objects," in *Proc. 2005 Texas Wireless Symposium*, Austin, TX, Oct. 2005, pp. 22–26.
- [4] S. R. Aroor and D. D. Deavours, "Evaluation of the state of passive UHF RFID: An experimental approach," *IEEE Systems Journal*, vol. 1, no. 2, pp. 168–176, 2007.
- [5] P. Bhartia, I. Bahl, R. Garg, and A. Ittipiboon, *Microstrip Antenna Design Handbook*. Artech House Publishers, Nov. 2000.
- [6] H. Kwon and B. Lee, "Compact slotted planar inverted-F RFID tag mountable on metallic objects," *Electronics Letters*, vol. 41, no. 24, pp. 1308–1310, Nov. 2005.
- [7] J. S. Chen and K.-L. Wong, "A single-layer dual-frequency rectangular microstrip patch antenna using a single probe feed," *Microwave and Optical Technology Letters*, vol. 11, pp. 83–84, Feb. 1996.
- [8] K.-L. Wong and K.-P. Yang, "Small dual-frequency microstrip antenna with cross slot," *Electronics Letters*, vol. 33, no. 2, pp. 83–84, Nov. 1997.
- [9] K. Kurokawa, "Power waves and the scattering matrix," *IEEE Transactions on Microwave Theory and Technique*, vol. 13, no. 3, pp. 194–202, Mar. 1965.
- [10] P. V. Nikitin, K. V. S. Rao, S. F. Lam, V. Pillai, R. Martinez, and H. Heinrich, "Power reflection coefficient analysis for complex impedances in rfid tag design," *IEEE Transactions on Microwave Theory and Technique*, vol. 53, no. 9, pp. 2721–2725, Sep. 2005.
- [11] M. Eunni, M. Sivakumar, and D. D. Deavours, "A novel planar microstrip antenna design for UHF RFID," *Journal of Systemics, Cybernetics and Informatics*, vol. 5, no. 1, pp. 6–10, Jan. 2007.
- [12] Ansoft Corporation, *HFSS Online Help*, Ansoft Corporation, Pittsburg, PA, 2006.
- [13] P. V. Nikitin and K. V. S. Rao, "Reply to 'Comments on "Antenna design for UHF RFID tags: A review and a practical application"'," *IEEE Transactions on Antennas and Propagation*, vol. 54, no. 6, pp. 1906–1908, Jun. 2006.
- [14] T. Instruments, *TI UHF Gen2 IC — Reference Guide*, 1st ed., Dallas, TX, Jun. 2006.

The University of Kansas



Technical Report

Unified SensorNet Architecture with Multiple Owners: An Implementation Report

Pradeepkumar Mani, Satyasree Muralidharan,
Victor S. Frost, Gary J. Minden, and David W. Petr

ITTC-FY2010-TR-41420-24

May 2010

Project Sponsor:
Oak Ridge National Laboratory
Award Number 4000043403

Copyright © 2010:
The University of Kansas
2335 Irving Hill Road, Lawrence, KS 66045-7559
All rights reserved.

Abstract

In practical sensor networks, it is possible that various components of the sensor network are owned and maintained by different organizations. This complicated scenario renders provision of security and management of these components as a challenging task. A framework for assured and controlled access for sensor networks is needed. In this paper, we describe such an architecture which integrates various component technologies into a unified framework. We also describe details of a proof-of-concept implementation, a standards - based multi-hop wireless sensor network application that demonstrates the salient features of our unified multi-ownership architecture.

CONTENTS

I	Introduction	1
II	Challenges in a Multi-Owner Scenario	3
III	Related Work	3
IV	Proposed Architecture	4
	IV-A Device Layer	5
	IV-B Repository Layer	7
	IV-C Application Layer	7
V	The ACE Architecture	9
	V-A Key Features of ACE for the Architecture for Sensor Networks	10
	V-B Access Protocol in ACE	12
VI	Demonstration of Multi-owner Unified Architecture	13
	VI-A Choice of Sensor	14
	VI-A1 Using the Nose	14
	VI-B Choice of Standardized Sensor Interface	15
	VI-C Choice of Sensor Network	16
	VI-D Sensing Application	16
	VI-E Proof-of-Concept Implementation	18
	VI-E1 Hardware Specifications	19
	VI-E2 Software Specifications	20
	VI-F Demonstration Procedure	20
	VI-G Summary of Experiences	21
VII	Conclusions	23
	References	24

LIST OF FIGURES

1	Proposed Unified Multi-Ownership Sensor Network Architecture	5
2	ACE Architecture showing the various components and their communications	10
3	Access Protocol showing the sequence of actions when an ACE Client contacts the Service Directory	12
4	Nose Server - Nose Communication via Multi-hop Wireless Mote Network	18
5	Sequence of Message Exchange between Nose Server and Nose	18
6	Various Components of Prototype Architecture	19
7	Demonstration Architecture showing Secure Access and Control of Sensor Capabilities in a Multi-Ownership Scenario	19
8	Demonstration Set up showing the various hardware components	21
9	User A successfully fetches profile from the Smell profile Database	22
10	User A successfully executes LOAD_PROFILE, but cannot run START_IDENTIFICATION	22
11	User B successful in executing START_IDENTIFICATION, but unsuccessful in executing LOAD_PROFILE	22

I. INTRODUCTION

SENSOR networks have been identified as being key technology in monitoring and detecting threats. These systems face critical technical challenges in providing a security and management architecture in scenarios representative of a large class of applications. Although the design and architecture of sensor networks [1], [2] and [3] have been studied and many networks have already been deployed [4], the development of a unified architecture for these systems when network elements are owned by disparate organizations is yet to be created.

As a motivating example, consider the following scenario, where a unique requirement is consideration of multiple owners of data and infrastructure: hazardous chemicals are often transported in trains. If the train meets with an accident, there is the possibility of a leak of one or more of the hazardous chemicals, which puts the inhabitants around the accident site at serious risk. It is very critical that the leaking chemical(s) and the extent of the leaks be identified as quickly as possible, so that evacuation procedures can be initiated in a timely manner, if required. A *monitoring team* (MT) could be assigned the task of identifying and monitoring the nature and extent of chemical leaks. Once the initial assessment of the MT is complete, a *containment team* (CT) can be assembled to contain the leak. The MT, at the very least, would require one or more chemical sensors deployed (either manually or through some other means) that can identify the nature and extent of the chemical leaks, and report the data to one or more authorized data collectors. To detect the presence of a chemical, the MT would probably need to upload (into the chemical sensor) an electronic profile of the chemical substance. This information could be obtained from a database of chemical profiles owned by the vendor e.g. the company initializing the transport of the chemical. In addition to the chemical sensors, there could be an assortment of sensing hardware composed of weather sensors, video feed, etc., that provide additional sensing data to the MT.

Each type of sensing equipment (including the database of profiles) could be owned by a different organization. Since the nature of the data could be sensitive, the MT would need to have the appropriate authorization to access the sensor data. The MT might also have to verify the authenticity of the data to counter the possibility of malicious attackers. The access level for the MT could vary by each device and database in this system. For example, the owner of the database of profiles could limit access of the MT only to those chemicals that are currently being transported in the train. In addition to having permission to read the chemical sensing data from the chemical sensors, the MT would also need to have write permissions to the chemical sensors to upload the chemical profiles. The MT would also need read permissions to access data from the weather sensor and video cameras. Once the nature and extent of

the leaks have been identified, a CT can then proceed to contain the leaks. The CT could be a different entity from the MT, and would require access to the sensing data from the assortment of sensors. The access permissions provided to the CT could differ from the ones provided to the MT. For example, the CT would not require write permissions to the chemical sensors, but may require write permissions to the camera to be able to tune the position of the camera appropriately.

Clearly, any authorization that is required needs to be acquired in real-time, spanning multiple administrative boundaries. This requirement can be circumvented by providing restriction-free access (super-user or administrative privileges) to the sensors and sensing data to any entity (MT and/or CT). Unfortunately, such a strategy is not only a blind approach, but could also be dangerous. For maximum flexibility and scalability, all sensors, data collection entities and databases need to expose standardized interfaces to be able to achieve seamless flow of data.

Simply placing sensors and cameras at the venue does not satisfy these requirements. Sensors must be integrated into an overall architecture to provide the responsible authorities both on- and off-site with situation awareness. Situation awareness includes: the location of personnel and assets (e.g., CT and health professionals), the state of the venue, and past experiences. A command center facility staffed from multiple agencies needs to be able to direct the collection of information to meet their immediate needs, translating the information into knowledge to be used as the basis of decisions on how best to maintain the safety of the venue. Information would flow to the headquarters facility from sensors, cameras, voice communications, and data archives, where each of these may be owned by separate entities. Situation awareness applications would facilitate transforming this information into knowledge. The decision-makers would then use this knowledge to direct the allocation of resources, e.g., point cameras or reposition personnel.

The remainder of this paper is organized as follows: We identify some key challenges in the development of a unified architecture for sensor networks with multiple owners in Section II. In Section III, we discuss related research. We discuss our proposed architecture in Section IV and in Section V we briefly describe the Ambient Computational Environment (ACE) [5] [6] architecture and its extensions that form the heart of our unified multi-ownership architecture. In Section VI, we describe, in detail, the prototype sensor network that we implemented to demonstrate proof-of-concept of our unified architecture; we also include some lessons learned in the process. Finally, in Section VII, we present the conclusions of our work.

II. CHALLENGES IN A MULTI-OWNER SCENARIO

The development of a unified architecture for sensor networks with multiple owners has not yet been fully explored and validated. Design, development, construction, deployment, and evaluation of a sensor network to enhance the safety and security pose significant research and technical challenges. From our experience with the prototype implementation of our proposed architecture, we are able to provide answers to the following questions:

- 1) What kind of access/control/security mechanisms need to be developed to facilitate the participation of multiple organizations? These mechanisms must allow for different policies for observing and/or controlling sensors.
- 2) What software systems and hardware are required to assist in the rapid and easy deployment, management, use, and redeployment of sensor networks?
- 3) To make the system affordable, how can commercial-off-the-shelf (COTS) wireless technologies be leveraged for use in an environment characterized by the heavy use of wireless communications equipment?

III. RELATED WORK

There is growing literature concerning the architecture and design of sensor networks [1], [2], [3], as well as the Open Geospatial Consortium Sensor Web Enablement (OGC- SWE) efforts [7] and Oak Ridge National Lab's SensorNet Information Architecture [8]. Several sensor networks have already been deployed [4]. Clearly, many of the component technologies required to realize the above chemical sensing scenario exist: sensors (especially chemical and radiological), cameras, communications systems and networks, data archives, GPS (or other location identification methods), GIS systems, and situation awareness applications. It would be a straightforward engineering task to design a "one-off" deployment, owned and controlled by one organization that may satisfy the needs of one venue. However, this is a point solution, and not a reusable one. A suitable integrated system architecture that could be reused across many venues is desirable in this case.

Our proposed Multi-Ownership Sensor Network (MOSN) architecture is an agents-based, tiered network architecture that supports internet connectivity, similar to the IrisNet architecture [9]. In addition, the MOSN is standards-based, and supports open standards such as OGC-SWE [7] and IEEE-1451 [10]. A premise of this research is that elements of the system will be owned by multiple organizations and communicate across administrative domains. Thus, there is a need for mechanisms that facilitate access to and control of sensors across multiple organizations as well as a requirement for rapid deployment.

Ownership by a wide variety of administrative domains is briefly mentioned in [9]. Unlike the IrisNet, the MOSN provides a secure system that facilitates the participation of multiple organizations in supplying needed component/subsystem functionality. Also, while SensorML [11] has sensor schemas that include security, user limitations and access constraints (like `documentConstrainedBy`), and schemas that identify the responsible party (like `operatedBy`), the integration of these into an overall system remains to be explored. A model of the new system has been implemented and evaluated. For a detailed comparison of the MOSN with existing sensor network architectures, the readers are referred to [12].

Our key contribution can be summarized as follows: we propose MOSN, a unified architecture for systems that have network elements owned by multiple organizations, which incorporates well-defined interfaces between different components with appropriate authorization/authentication mechanisms that are secure and suited for disseminating and analyzing sensor information. We also demonstrate the salient features of our architecture through a proof-of-concept implementation consisting of a standards-based, multi-hop wireless sensor network application. To the best of our knowledge, there is no other architecture that addresses the unique issues of multiple ownership in a sensor network.

IV. PROPOSED ARCHITECTURE

The objective here is to develop a unified architecture that has elements owned/controlled by a variety of organizations which can communicate across across administrative domains. Our proposed architecture is general (not a point solution), scalable (in size and evolution of technologies), flexible (able to mix and match technologies based on the venue requirements), economical (based on COTS technologies), and leverages standards where possible. The proposed approach facilitates multiple organizations providing different services, enabling the development of a business model based on sensor network technologies. The key features of the proposed architecture include assured and controlled access to sensor nodes in a multi-owner environment, archiving and information dissemination.

The architectural components are divided into three layers as shown in the Fig. 1 based on their functionality:

- 1) Device Layer
- 2) Repository Layer
- 3) Application Layer

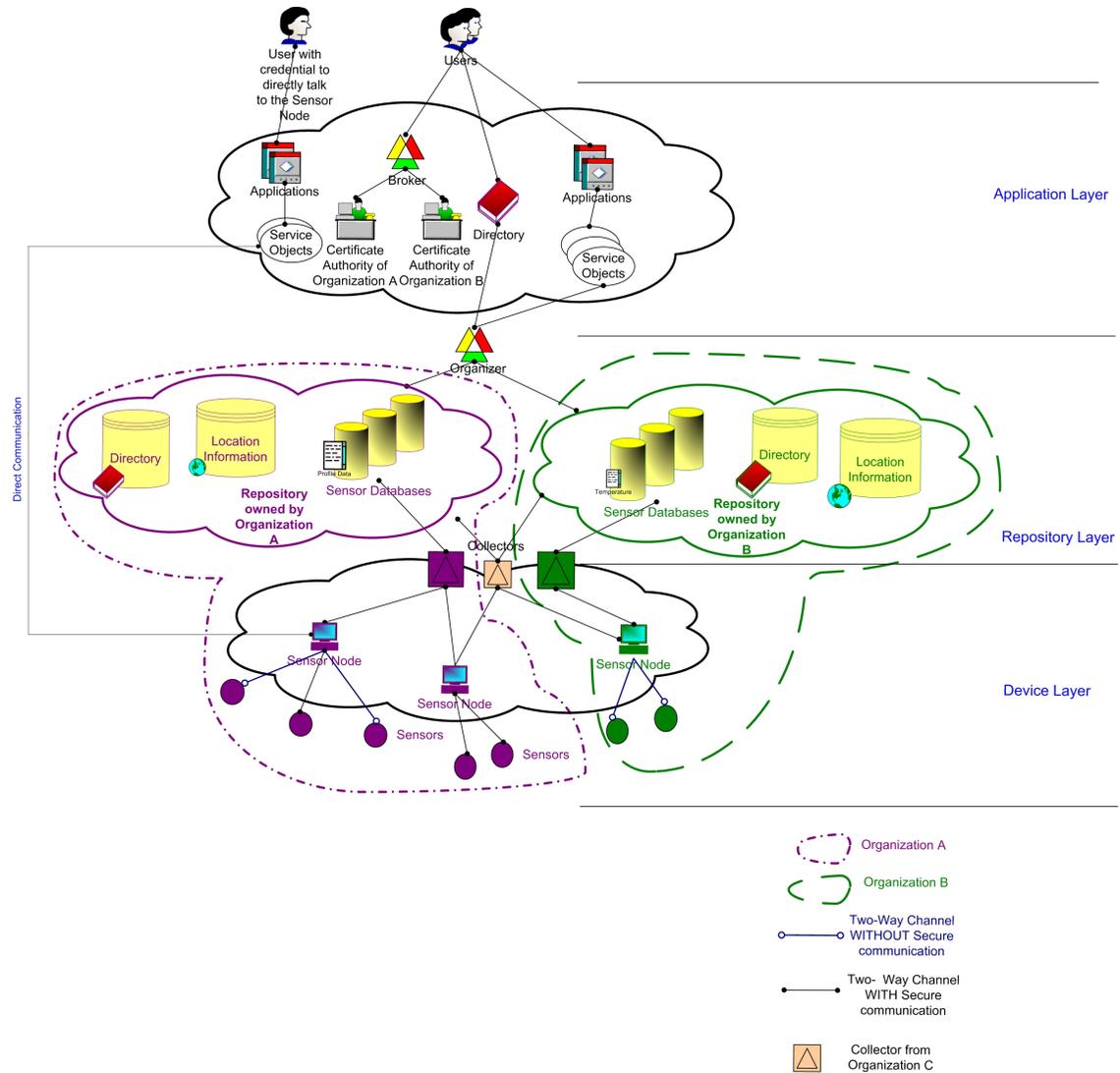


Fig. 1. Proposed Unified Multi-Ownership Sensor Network Architecture

A. Device Layer

Device Layer comprises all the physical sensor endpoints together with the first level of data access and management points for the entire architecture. This consists of:

- Sensors
- Sensor nodes
- Sensor services
- Collectors

A *Sensor* is a device that responds to an environmental quantity (e.g. light, temperature, etc.) by generating a functionally related output usually in the form of an electrical or optical signal. Sensors communicate the collected data to the node that controls them in the sensor network. The sensors could be of different types such as radiological, mechanical, optical or chemical sensors. Often sensors are characterized by small size and low energy consumption. Sensors can be broadly classified into two types: *active sensors*, and *passive sensors*. Active sensors usually engage in two-way communication with the data collecting host. They can accept commands from the sensor node in real time, and send appropriate responses back. Passive sensors, on the other hand, simply send back (periodic/event-driven) data to the collecting node. They usually do not support a *Request- Reply* type of communication with the control node.

A *sensor node* is a computer that typically manages one or more sensors through a set of services. The sensors could be directly connected to the sensor node either through serial or parallel ports or through a multi-hop network. The communication between the sensors and the node may or may not incorporate a secure communication. The security of this link depends upon the nature of the connectivity. If the sensors are plugged to the node directly through serial or parallel ports, then the communication is inherently secure. With a multi-hop wireless network, security should be explicitly incorporated in the communication links.

Sensor services are programs that control the sensors attached to the node. There could be one or more services per node, with each service dealing with one sensor. In this proposed architecture, we assume that each sensor service (program) controls exactly one sensor. In practical applications, a single service could control more than one sensor. The architecture proposed here could be extended so that one service supports multiple sensors.

Collectors are programs that collect data from these services and transport them to the repository layer for further use. There could be one or more collectors depending on the number of devices in the device layer. The communication between the collectors and the sensor services follow the access control mechanism discussed later in this document. Collectors must authenticate and authorize themselves with the service, before tasking or configuring a sensor. Collectors gather data in one direction (from device to repository) while the sensor services load data or commands in the other direction (from repository to device) e.g., load commands from the Sensor Databases to the sensors. Collectors talk to the devices which typically belong to their organization or domain; our solution is not restricted for such a communication but spans across different organizational domains.

B. Repository Layer

This forms a link between the lower device and the upper application layer allowing dissemination of information. This consists essentially of databases of two types:

- Infrastructural databases that are mandatory and store information required to support the system:
 - Service Directory - database of current services available such as Temperature Sensing Service, Chemical Sensing Service.
 - Regional Database - database of location of sensors.
- Sensor Databases that store and retrieve sensor data, *e.g.*, database of images captured by cameras used for surveillance.

There could be multiple repositories (sensor and infrastructural databases) within this layer, each owned by a different organization. Within a given administrative boundary, the services from the device layer register themselves with the Service Directory when they come online, necessitating each organization to maintain a list of currently available services. This is illustrated in Fig. 1, which shows two organizations, A and B, with their corresponding administrative boundaries. Each organization has its own set of devices and repositories. Each organization might implement redundancy in its repositories and devices for fault tolerance and robustness.

C. Application Layer

The application layer provides a unified view of the various components of the architecture to the user. A *user* in this architecture is a human being who uses the infrastructure for various applications. *Applications* are programs that can either talk to the organizer to get the processed data or talk to the services directly, and comply with the open standard implemented by the system. An *organizer* is a program that fetches data from the repository layer and presents meaningful interpretations to the requesting application. Consider the chemical sensing application described in section I. The MT at the accident site could have temperature sensors as part of the weather sensors deployed to monitor various threats such as fire, etc. An agent program collects raw temperature data, converts them into location-centric values, and writes them into the database of temperature values. The system could also have cameras deployed at various locations in the venue to provide on-demand, live video feed. The temperature sensors, cameras and the database could be owned by different organizations. A security officer has a monitoring application program that shows the location of the various temperature sensors, and the temperature values recorded by them (retrieved from the database). The monitoring application

raises an alarm whenever any sensor shows a temperature value that is outside a specified range. The security officer uses the monitoring application to request video feed for the location of the suspected fire. Based on the contents of the video feed, the office can then initiate appropriate actions.

The architecture described here could be applied directly to this scenario. Referring to the architecture, the agent program in this example plays the role of the collector and organizer- it transports sensor data (temperature, video) from the device layer to the repository later, and from the repository layer to the application layer (monitoring application program used by the inspector). The owner of each of these components (sensor, camera, database, etc.) is an *organization* and the security officer is the user.

This 3-tier architecture is layered with organized communication between the layers using the intermediaries such as collectors and organizers. However, we anticipate that some scenarios might require a user talking directly to a device without having to pass through this layered architecture.

Consider a situation where the user takes direct control over the sensors of all organizations and may wish to control them without having to talk to the organizer or the collector. In such a case, the user will need authorization to interact with devices from all organizations. The user will use the applications to talk to the sensor services controlling the devices through an out-of-band communication. Our solution also provides a way to have this *Direct Communication* between the user and the devices as in Fig. 1. With reference to the chemical sensing security system described above, the video feed from the camera requires direct communication between the user and the camera.

The description of the unified architecture (MOSN) is not yet complete. We still have not discussed the following issues:

- 1) Inter-layer communication
- 2) Policies for the following:
 - a) Secure, controlled and authorized access to a specific component (sensor, database, etc) in a multi-owner heterogeneous sensor network. These policies will be important if we would like to restrict access to only a specific instance of the component, in the presence of multiple similar components (e.g. temperature sensor # 43)
 - b) Secure, controlled and authorized access to a specific functionality of a specific component in a multi-owner heterogeneous sensor network. These policies are required if we would like the client to have access to only a subset of functionalities offered by a specific component. (e.g. only READ function from Temperature Database, but no WRITE privileges)
- 3) Propagation and enforcement of these policies

We address all the above-mentioned issues by borrowing components from the ACE [5], [6] architecture. The device control and data flow mechanisms developed for ACE are used here to manage/control connections between applications and sensor nodes. The ACE control mechanisms provide for authentication by the device of the controlling application, authorization to access and control the device based on an established security policy, confidential transmission, and integrity checks. The ACE data flow mechanism supports real time exchange of data between applications and devices that is private and checked for integrity. ACE supports establishing services within the environment to archive data flows, replicate data flows to multiple receivers, and play back archived data. Since the ACE architecture is a key component in the MOSN, the next section discusses the ACE architecture in some depth.

V. THE ACE ARCHITECTURE

ACE provides a secure communication fabric between the various components in the various layers of the unified architecture. The ACE architecture was developed to be the basis for a pervasive system, where the users have long-lived workspaces and mobility within the environments irrespective of rooms or machines. In other words, in the ACE system, the users can roam anywhere, while still preserving their sessions with the resources.

ACE supports two types of communication channels between the client and the service:

- 1) *Control channel*: Provides a way for communicating control messages. It is a reliable in-order channel.
- 2) *Media channel* : Provides a way for communicating audio and video. Reliability and in-order delivery are not important in this channel.

In the ACE architecture, *services* constitute the atomic level of computation. The most important services are the three core services - *Service Directory service*, *User Database service*, and *Regional Database service*. These core services are programs that inter-operate as shown in the Fig. 2, performing user authentication and verifying user authorization to allow a client to only access resources that he is permitted to. The *Service Directory* is a directory service that locates all available services as well as their characteristics (Name, Location, and Service Class). All services register and un-register with this service. Since this is the directory for all the other services, the location of this service is fixed. The *User Database* is a database of all users in the system. The information includes Public Key, Name, Login name and Login characteristics. In ACE, the login characteristics include information like passwords, finger prints and iButton identifiers that can identify the user. The *Regional Database* is a database of the

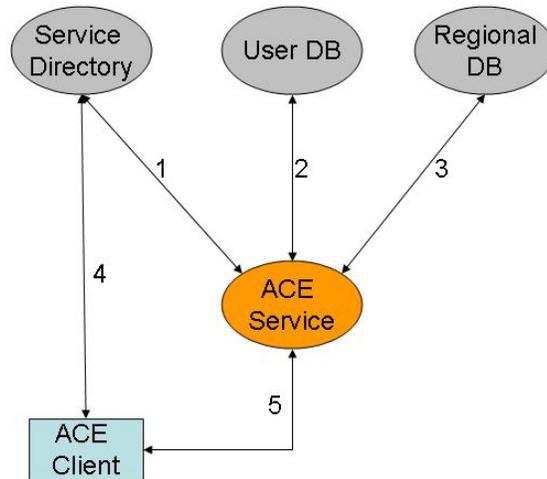


Fig. 2. ACE Architecture showing the various components and their communications

information of all the service locations in the system (e.g. rooms containing various sensors, geographical co-ordinates of the accident site and associated sensors, etc.).

A. Key Features of ACE for the Architecture for Sensor Networks

We have already mentioned that the architecture for sensor networks demands *secure communication* between the various components of the system (client, services, etc.), *component-level* access control to each and every component in the network, and *method or function-level* access control to every functionality of a given component. We briefly discuss the features of ACE that are suitable to the requirements of this architecture.

1) Client Server communication using Enhanced RMI:

Whenever a client wants to talk to the Service, the client provides his credential showing that he has permissions to talk to the service to access the resources. The service validates his credential before providing the resource. *The authorization not only provides access to the resource, but also extends to every method that the client requests to perform on the device.*

The services present themselves as Java remote objects. The functionalities that the services advertise are given in the Java Interface. The client obtains the remote object to the service and performs actions using the Java RMI. This feature can be directly applied to the target architecture with the collectors being the clients, sensors being the devices, and the sensor service being the gateway between the collectors and the sensors.

2) *Secure communication using TLS:*

Transport Layer Security (TLS) provides authentication of the user and security of the message exchanges in the control channel. The users are identified by public key of the asymmetric key (either RSA or DSA). The public key is certified by a Certificate Authority to verify the validity of the key. A Certificate Authority (CA) is an entity that issues digital certificates. Each organization will have its own CA to issue certificates for users within that organization. The role of the CA is to issue 1) certificates to identify the users and, 2) credentials to identify the actions that can be performed by the users.

If a user wants to talk to devices from multiple organizations, then he/she needs to contact the CAs of different organizations individually to get certificates. The user presents this signed certificate to any service for authentication. At the end of the handshake between the client and the server, a session key is negotiated and all the messages are encrypted with this key using any symmetric key algorithm such as AES or DES. In this architecture, security of the message exchanges between the collectors and the sensor services is important. This TLS and AES encryption of the ACE framework provides authentication of client-server, secured communication by encrypting the message exchanges and provides error-free delivery as required here.

3) *Access Control using KeyNote Trust Management System:*

Once user authentication is complete, the service programs determine the actions that the user can perform based on the exact permissions assigned to the user - i.e., the service must authorize the actions requested by the user based on these permissions. ACE uses KeyNote Trust Management [13] to provide access control on the actions requested by the users. KeyNote provides a simple language for describing and implementing security policies, trust relationships and digitally-signed credentials to control potentially dangerous actions over untrusted networks. The policies are specified using the KeyNote language, which are then signed by the CA, and given to the client as a credential file. The user presents this credential file to any service that he wishes to access.

This trust management allows the system to control access to the actions performed by a collector on the sensors. Each method that the client is trying to access through the remote object of the service can be checked for authorization by querying the KeyNote engine. If the collector does not have a valid credential, it cannot perform the requested action on the sensor and an exception is raised. Since this authorization is implemented within the service infrastructure, all services inherently implement the authorization procedure. Detailed descriptions of the KeyNote trust management system and the KeyNote description language are beyond the scope of this paper. Interested readers

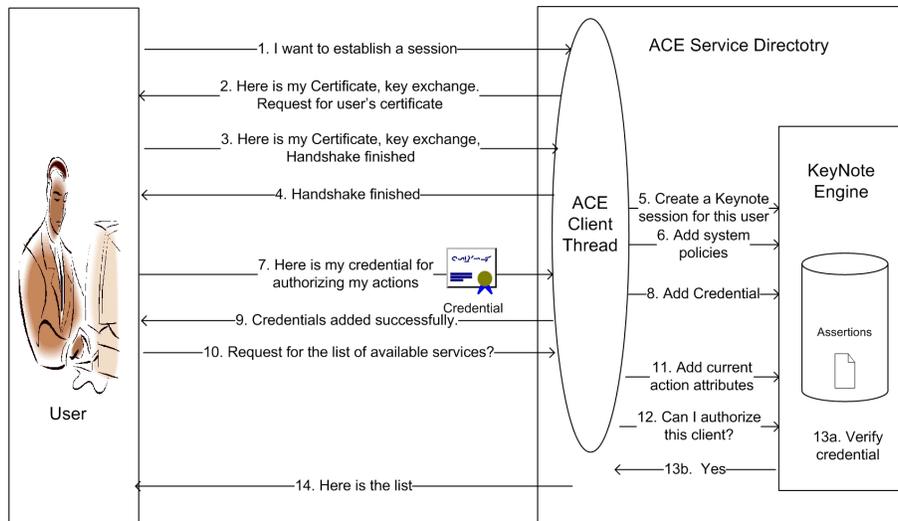


Fig. 3. Access Protocol showing the sequence of actions when an ACE Client contacts the Service Directory

are referred to [13] and [14] respectively.

B. Access Protocol in ACE

In our architecture, the ACE framework provides the required client-server authentication, secure communication (by encrypting the message exchanges), and error-free delivery. The ACE architecture contains the KeyNote mechanism, which enables formulation and enforcement of policies.

The user presents the signed certificate from the CA to any service for authentication. The sequence of actions when an ACE user talks to an ACE Service is described below. Each time a new client contacts a service, the service spawns a new thread dedicated to the communication with the specific client. Typically the following happens when a client wishes to access a resource, as shown in Fig. 3:

- 1) User contacts the service (the client thread of the service) for establishing a session.
- 2) The service replies with its certificate for authentication, key exchange for establishing session key and requests the user for his certificate.
- 3) The user replies with his certificate and session key exchange. The user verifies the Server's certificate. The client sends *Finished*.
- 4) The service contacts the user database to verify the user's certificate. Once verified, the service sends *Finished*. The TLS authentication is completed and a session is established.
- 5) The service creates a new KeyNote Session that will be used for user's authorization.

- 6) The service provides the required policy to KeyNote database to be used to verify the client's credentials later.
- 7) The client provides his credential to the service.
- 8) The service adds this credential to the KeyNote database.
- 9) The service replies the result of adding the credentials to the user.
- 10) The client requests the service to list the available services.
- 11) The service provides to KeyNote the current set of action attributes such as domain in which this application is used, room in which the service runs, current Time and the method requested by the client.
- 12) The Service queries KeyNote for authorizing the action requested by the client.
- 13) The KeyNote engine verifies the credential against its current set of attributes and returns the result to the Service.
- 14) The Service determines whether to perform the action requested by the client or not depending on the result from the KeyNote engine. If the Service performs the action, it returns the result to the client. Else, it returns an *Access Denied* exception to the client.

VI. DEMONSTRATION OF MULTI-OWNER UNIFIED ARCHITECTURE

We wanted to choose a simple, yet powerful application to demonstrate the salient features of our multi-owner unified sensornet architecture, namely secure access and control of the various entities in a heterogeneous network of sensors. To this end, we chose a simplified version of the chemical detection sensing application described in Section I. Initially, a database was populated with profiles of the various chemicals of interest. An authorized client can retrieve the profile corresponding to a specific chemical, and load it into the detection sensor. Another authorized client (or perhaps the same client as the initial one) can issue commands enabling the sensor to detect the presence of a chemical. Note that the database (and perhaps the individual profiles themselves), and the sensor *need not* be owned by the same organization.

To build this system, we had to choose a number of components. First, we had to decide on the specific sensor we were going to use. Based on the requirements of the sensor (bandwidth, processing power, etc), we then had to choose a wireless sensor network technology to build the network. For future extensibility with minimal system reconfiguration, we chose a standard sensor interface. All the hardware and software components associated with the sensor, the wireless network and the software library that provides the standardized interface belonged to the Device Layer. To support this architecture, we needed databases for the core services. We also needed databases that stored the sensor-related data (e.g. chemical

profiles, video, etc.). These databases formed part of the Repository layer. Finally, we needed application programs that allowed a client to interact with the components in the system. This software was part of the application layer. Below, we briefly describe our choices for the various components, and some issues that we faced with our choices.

A. Choice of Sensor

To realize the application, the first task at hand was to choose a sensor that was easy to operate and program, while at the same time had a sufficiently rich set of features that would allow us to demonstrate secure control and access under a variety of multiple ownership scenarios. It is apparent that we would require an active sensor for the application under consideration. We chose the Cyranose 320 Electronic Nose sensor [15] (referred to simply as Nose henceforth), which met all of our requirements. The Nose is a handheld sensing device that can be trained to identify the presence of certain chemical compounds or substances.

1) *Using the Nose*: The training procedure is a rigorous process, and has to be conducted in controlled environments. The training procedure consists of a series of controlled exposures of the Nose to the target substance. Each exposure results in a smell print, and the result of the training procedure results is a series of smell prints, collectively called as a smell profile (corresponding to the target substance). The Nose stores the smell profile in its internal memory. This smell-profile can be retrieved from the Nose, stored externally in the form of a file, and can be loaded into the Nose at a future date. The identification procedure requires the Nose to be exposed to unknown substance for a brief period of time. The Nose compares the smell-print that is created due to this exposure against the smell profile (series of smell prints) already stored in the Nose. The result is a percentage match between the unknown substance and the known substance (the substance corresponding to the smell profile loaded into the Nose). During the training process, we realized the following:

- The accuracy of the smelling process depended heavily on the accuracy of the training process.
- The Nose is not a suitable device for real-time scenarios (such as smoke detection, etc.) due to the time to generate results. It is targeted as a hand-held device that requires operation in a controlled environment.

Despite these disadvantages, the Nose was still the preferred sensor to demonstrate proof-of-concept of the architecture due to its simplicity of operation and rich set of control features. For our experiments, we trained the Nose to distinguish between *Isopropyl Alcohol* and *Water*. The following subset of commands were the focus of the demonstration:

- 1) LOAD_PROFILE - loads the provided smell profile from the file provided
- 2) START_IDENTIFICATION - starts to “smell” the sample provided to match it to one of the possible candidates in the smell profile
- 3) FETCH_RESULTS - returns the result of the latest identification along with a confidence measure

B. Choice of Standardized Sensor Interface

To support a heterogeneous mix of sensors, a standardized interface was necessary to communicate with the sensors. Such standards ensure that a standardized software interface will be exposed by the architecture to the sensors/devices in the system. These interfaces minimize developmental efforts required when new sensors need to be integrated into the system in the future.

The IEEE 1451 standard [10] is one such standard that has been developed precisely for this purpose. Also the ORNL SensorNet architecture [8] has selected the IEEE 1451 standard for this effort and its services. The IEEE 1451 standard was also chosen as an example for the prototype architecture - the system designers could use any competing standard such as Microsoft’s Universal Plug and Play (UPnP) [16], or Optical Sensor Interface Standards [17] in case of optical sensors. We do not make any recommendations on the choice of the standard. For more details on the IEEE 1451 standard, readers are referred to [10].

A key challenge in this task was to efficiently integrate the developed framework and IEEE 1451, so that standardized interfaces are exposed to the sensors by the Nose Server. An IEEE 1451-compliant server is also called a Network Capable Application Processor (NCAP)[18]. We chose the Java Distributed Data Acquisition and Control (JDDAC) library [19] to build our custom Nose NCAP. Though the JDDAC was not 100% compliant with IEEE 1451, it was deemed sufficient due to lack of alternative IEEE 1451 software in Java (Java was preferred because the bulk of the ACE code was written in Java), and also to demonstrate proof-of-concept.

One of the main obstacles in using the Nose as a sensing device in a IEEE 1451 environment was that the Nose was not an IEEE 1451 - compliant device. We overcame this obstacle by placing a Nose-1451 interface in the NCAP [18] module, that did the following:

- 1) convert incoming IEEE 1451 compliant messages from the client to Nose-Specific commands and send it to the Nose, and
- 2) convert incoming Nose-specific message from the Nose into a IEEE 1451 message, and send it to the client

In addition, the JDDAC library did not support a *Request-Reply* type of communication, which is critical in future sensor networks, and thus to our application. We overcame this by extending the JDDAC IEEE 1451 library by implementing the necessary features from the IEEE 1451.0 [18] standards document to support a *Request-Reply* type of communication.

C. Choice of Sensor Network

The next step in our research was to set up a wireless sensor network to study and demonstrate secured and controlled access to sensors with multiple owners. After investigating some sensor network technologies, we decided to construct the wireless network using MICA2 motes developed by Crossbow Technologies [20]. The following were the salient features of the MICA2 motes:

- *Processor*: Atmel ATmega 128L MicroProcessor (7.37 MHz clock)
- *Memory*: 4KB data RAM, 128KB Program Flash, 512 KB (serial) Flash
- *Radio*: ChipCon model CC1000 multi-channel transceiver (868/916MHz, 433 or 315MHz)
- *Data Rate*: Up to 38.4 Kb/s
- *Range*: ~30m (indoors), ~150m(outdoors)
- *External Interface*: 51-pin expansion connector
- *Power Source*: 2 AA batteries
- *Operating System*: TinyOS (TOS)

The popularity of motes and TinyOS in the sensor network community and a rich set of libraries that facilitates quick application-building were the primary motivations for selecting them. For high-bandwidth applications like video, the motes-based sensor network would not be a good choice. In such cases, one could use a 802.11-based wireless mesh network.

D. Sensing Application

We developed an application using NesC in the TinyOS platform [21] that would manage message transfer between the Nose and the Nose server via multiple hops of motes. The Nose was directly connected to the destination mote via a serial port connection. TinyOS supplies a serial port communication module that uses a framed message format, which would not be understood by the Nose. This complication was overcome by writing a custom serial port communication module that transfers raw data across the serial port without any framing, using low-level TinyOS routines.

Another problem with TinyOS was that the default message size was 29 bytes, while the message sizes generated by the Nose was of the order of few hundreds of bytes, sometimes even greater than the largest

message size supported by TinyOS (128 bytes). So, we decided to fragment the large Nose messages at the source to fit the default TinyOS packets, and reassemble the fragments at the destination. Fragmentation-Reassembly could be implemented by increasing the default message size to the maximum message size (128 bytes). However, TinyOS messages were the de facto mode of message exchange between the various layers in the TinyOS stack, and thus increasing the default message size was not viable because it resulted in an overall RAM requirement that exceeded the available RAM capacity. Given that the application was not particularly time-sensitive or bandwidth intensive, there was no need to increase the size of the fragment. Fragmentation/Reassembly (FR) was thus implemented as follows, using the default TinyOS message size (29 bytes) as the maximum fragment size:

A fragmentation layer was introduced between the application layer and routing layer of the source, and a reassembly layer was introduced between the application layer and routing layer of the destination. The FR module was designed in a very modular fashion, so that any routing protocol can be “plugged” into the application. A simple stop-and-wait protocol with positive acknowledgement was introduced for error recovery, in the event that a fragment was lost in transit. If an ACK is not received at the source within a dynamically determined time frame, the fragment is assumed to be lost, and a retransmission procedure is initiated. The number of retransmissions is limited to a predetermined maximum to recover from a loss of route to the destination.

Multi-hop communication in our application was made possible by using TinyAODV [22] as the routing protocol. TinyAODV is derived from the well-known Mobile Ad hoc NETWORKing (MANET) [23] routing protocol, the Ad hoc On-demand Distance Vector (AODV) [24] routing protocol. TinyAODV retains only a subset of AODV’s features so that it can fit within the memory constraints imposed by the motes (and thus *tiny*). TinyAODV was chosen because it was simple, and it came bundled with the library supplied by Crossbow.

Fig. 4 shows the multi-hop communication schematic between Nose Server and the Nose in the Nose Service application. With reference to our architecture, the Nose Service plays the role of the collector, while the motes network and the Nose sensor form part of the device layer. We only used the following three commands: `LOAD_PROFILE`, `START_IDENTIFICATION`, and `FETCH_RESULT`. These commands were issued by the Nose service and were transmitted to the Nose via the multi-hop wireless motes network. Fig. 5 shows the sequence of message exchanges between the Nose Server and the Nose.

The following section describes the use of a proof-of-concept prototype, and the resulting lessons learned.

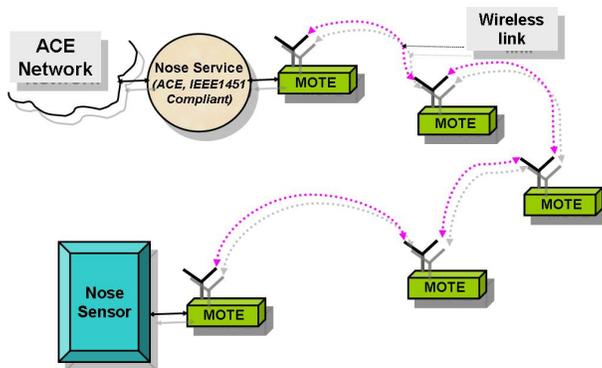


Fig. 4. Nose Server - Nose Communication via Multi-hop Wireless Mote Network

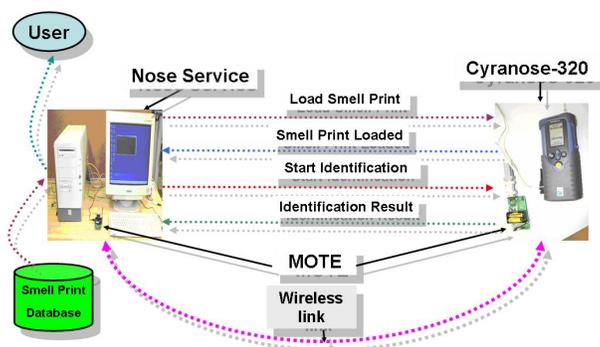


Fig. 5. Sequence of Message Exchange between Nose Server and Nose

E. Proof-of-Concept Implementation

Fig. 6 shows the various components of the prototype unified multi-owner sensor network architecture. There are no performance measurements associated with these demonstration experiments. Instead, the objective of our prototype implementation was to demonstrate the interaction of various entities (owned by multiple vendors) in a secure, controlled-access environment.

Fig. 7 shows the schematic of the multiple-ownership architecture. Two clients, user A and user B are used here, and the clients were each assigned different permissions to access and control the Nose. The client programs of both users were run on the same machine. We had server-1 running the ACE Core Services: the ACE Service directory, ACE User database service, and the ACE Regional database service. To demonstrate that the various components of the architecture could function in a distributed manner, we ran the Nose (chemical sensor) service on server-2. The nose server, which was built to IEEE 1451 specifications, was connected to an electronic nose via a multi-hop MICA motes network. Here,

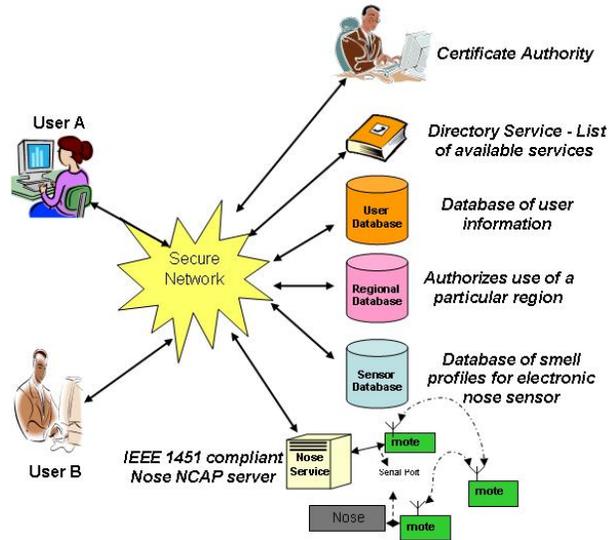


Fig. 6. Various Components of Prototype Architecture

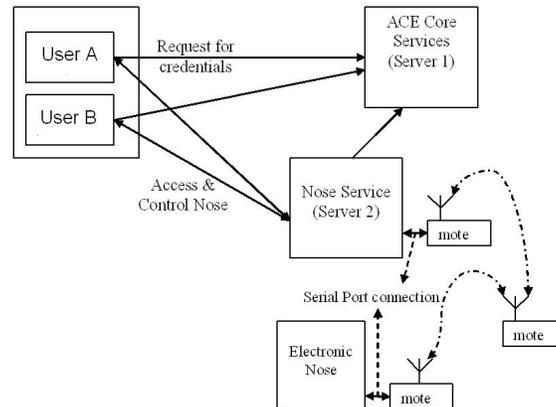


Fig. 7. Demonstration Architecture showing Secure Access and Control of Sensor Capabilities in a Multi-Ownership Scenario

we showed that the clients could successfully execute only the commands that they were permitted to execute on the Nose, and were not able to execute commands for which they did not have the necessary permissions. Fig. 8 shows the actual demonstration configuration.

1) *Hardware Specifications:*

- *Server 1-* Pentium III 750 MHz processor, 256 MB RAM, 10 GB HDD, Red Hat Enterprise Linux WS release 4
- *Server 2-* Dell Latitude D820, Pentium Dual Core T2500 2.0 GHz, 1 GB RAM, 80GB HDD, Fedora

Core 5 (2.6.16)

- *Client Host* - Pentium III 933 MHz processor, 512 MB RAM, 80 GB HDD, Red Hat Enterprise Linux WS release 4
- *motes* - MICA2 motes, 915 MHz, programmed using nesC (TinyOS 1.1.7) on a mib510 board (serial port). The transmission power level of the motes was reduced by 20 dB to reduce the range of the radios to ~5 ft, to demonstrate multi-hop routing.
- *Electronic Nose* - Cyranose 320, serial # B001203158, mounted on a chemistry lab stand

2) *Software Specifications:*

- *Server Application* - The server application or the Nose service could issue commands to, and receive responses from the Nose. It contained the Nose NCAP mentioned above, so that all communication between the client, server and sensor was IEEE 1451 compliant. We used the JDDAC library, which has been built around IEEE 1451 specifications, to build our Nose NCAP. It involved designing a new Function Block to process and issue Nose-specific commands and a Transducer Block to provide serial port IO capability. The ACE communication channels provided the communication fabric between the client and the server. The Nose server exposed standard ACE interfaces to the client, so that the Nose service could leverage the ACE security, authentication and authorization mechanisms that are critical to our architecture.
- *Client Application* - The sole purpose of the Nose Client was to issue IEEE 1451-compliant commands to the server, based on the user input. The client application was a simple GUI which simply lists the commands that can be executed on the Nose. To ensure inter-operability with ACE, all client applications were written in Java and used Remote Method Invocation (RMI) [25] to communicate with the sensor services. With JAVA-RMI, the services present themselves as remote objects. The applications get handles to these remote objects and use them to talk to the services. As recommended in the JDDAC documentation [19], Eclipse IDE was used to build the Client and Server applications. The version of Java that we used was JDK 1.5.0.8.

F. *Demonstration Procedure*

As mentioned earlier, here we model each of components of the network (Nose, Smell profile Database, etc.) as belonging to a different organization, and the clients were not necessarily associated with any of these organizations. This lends itself to a truly multi-ownership scenario, where the clients require access to resources that span multiple administrative boundaries. The clients already possess the necessary authorization (credential files). We do not discuss how each client obtained the credential files from an

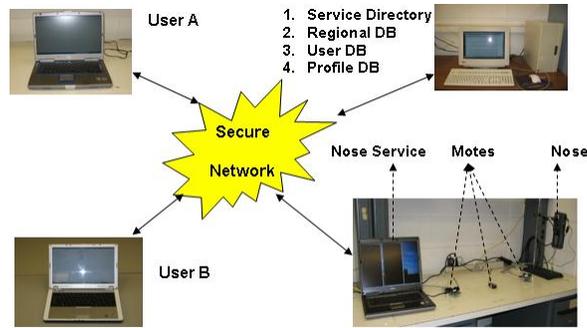


Fig. 8. Demonstration Set up showing the various hardware components

appropriate CA - we simply assume that the authorization was obtained via some legitimate method. The following were the permissions assigned to each user:

- 1) *User A*: `LOAD_PROFILE`,
- 2) *User B* `START_IDENTIFICATION` and `FETCH_RESULT`.

In addition to these permissions, the credential file contains other relevant information such as the time frame within which the user is permitted to execute these commands, the room that contains the Nose (regional information), etc.

To execute `LOAD_PROFILE`, first a profile had to be fetched from the smell profile database (repository layer), and then loaded into the Nose. Using the simple client GUI provided (client application), user A was able to successfully retrieve the file corresponding to "Isopropyl Alcohol - Water" profile from the database of smell profiles. Fig. 9 shows the sequence of associated message exchanges that leads to fetching the smell profile. As a second step, user A could successfully load the smell profile into the Nose by executing the `LOAD_PROFILE` command. User A then tried to execute `START_IDENTIFICATION` command, but received an "Access Denied" error message due to lack of permissions. Fig. 10 shows the sequence of message exchanges that lead to successful execution of `LOAD_PROFILE`, and an unsuccessful attempt in executing `START_IDENTIFICATION`.

G. Summary of Experiences

The above-mentioned demonstration validates the design of the proposed architecture. Our experience with the prototype implementation provided a better understanding of some of the issues that we listed in Section II. Lessons learned include:

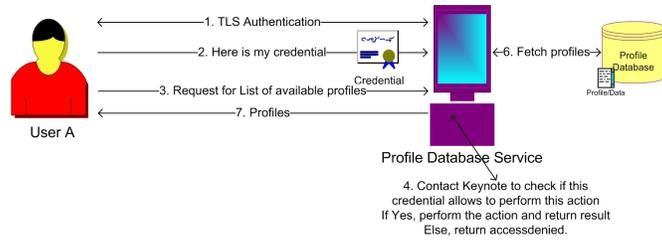


Fig. 9. User A successfully fetches profile from the Smell profile Database

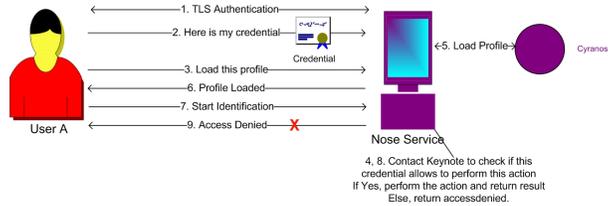


Fig. 10. User A successfully executes LOAD_PROFILE, but cannot run START_IDENTIFICATION

- Scalability:* The architecture does not require the control components to be co-located with one another or with the devices, and functions in a very distributed fashion. Thus, the architecture is scalable. However, we have not conducted any formal studies to assess the scalability of our architecture, and such evaluation is definitely an area of future work.
- Security/Access/Control Policies for Multiple Organizations:* The KeyNote trust management system turned out to be an excellent choice for formulating and enforcing security, control and access policies for components belonging to different organizations. However, the policy specification language lacked structure and proved to be cumbersome when specifying policies for a large number of heterogenous sensors, each with multiple functionalities. A new policy specification language, with

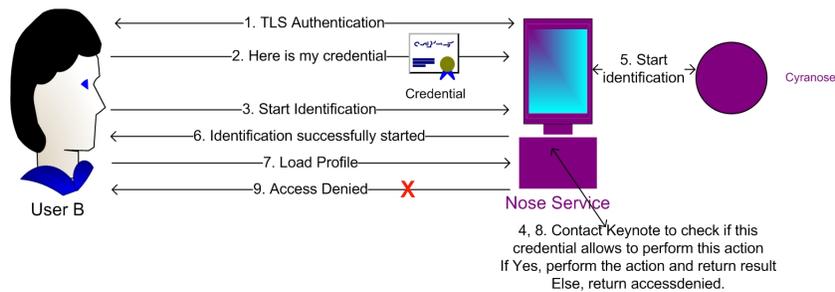


Fig. 11. User B successful in executing START_IDENTIFICATION, but unsuccessful in executing LOAD_PROFILE

a hierarchical structure is being considered to overcome this problem.

- *Software Support:* The ACE architecture software is perhaps the most critical software component of the system. ACE provided authentication, authorization and secure communication services. The other major software component was the IEEE 1451 library. IEEE 1451 enabled future extensibility of supported sensors with minimum software developmental effort. However, integrating the IEEE 1451 library into the ACE architecture required significant effort. We expect that similar effort levels will be required if we wish to integrate other sensor interface standards into the system. A convenient improvement will be to enhance the system to support these standards as simple plug-in modules, so that deployment of this architecture is simplified and rapid.
- *Device Reconfiguration:* The use of a standardized interface to communicate with the sensors greatly reduces the need for device as well as system reconfiguration. The IEEE 1451-compliant sensors are also called smart transducers - they have the ability for self-identification, and can be configured during service start-up. Sensors can be reconfigured, added or removed from the system with great ease.
- *Device Naming/Addressing:* The use of the regional database in the developed architecture ensures that the addresses or names assigned to the devices only have local significance. This greatly simplifies the addressing or naming of the devices. If the sensors are moved to a different location, then the corresponding entry in the regional database will be updated, and (possibly) the device will be assigned a new address.
- *Hardware Support:* For our implementation, we entirely used COTS wireless sensor components (motes), which are not only affordable, but also easy to deploy. The limiting factor of motes is that it requires programming using NesC in the TinyOS platform. A more attractive alternative could be to use SunSpots [26] or gumstix [27] that allow for programming in Java. We are currently exploring these technologies as possible wireless sensor network building components.

VII. CONCLUSIONS

In this paper, we described a sensor network architecture which integrates various component technologies into a unified framework that is rapidly deployable, scalable and owned by variety of organizations. We also created a proof-of-concept implementation, which is a standards-based multi-hop wireless sensor network application. In our framework, the control mechanisms provide for authentication of the client (by the device of the controlling application), authorization to access and control the device based on an established security policy, confidential transmission, and integrity checks. The data flow mechanism

supports real time exchange of data between applications and devices that is private and checked for integrity. Security policies are specified and enforced via a KeyNote Trust Management System. This prototype demonstrated the salient features of our unified multi-ownership architecture, namely assured access and fine-grained control to the various components in the system across multiple administrative domains. In particular, we showed that the users could access and control resources (devices, functions of devices) belonging to multiple owners, provided they had the necessary authorization to do so. Unauthorized users were denied access to the resources.

REFERENCES

- [1] A. Hac, *Wireless Sensor Network Designs*. West Sussex, England: Wiley & Sons, 2003.
- [2] D. Estrin *et al.*, "Connecting the Physical World with Pervasive Networks," *IEEE Pervasive Computing*, pp. 59–69, January–March 2002.
- [3] I. F. Akyildiz *et al.*, "Wireless Sensor Networks: A Survey," *Computer Networks*, vol. 38, pp. 393–422, September 2002.
- [4] R. Szewczyk *et al.*, "Lessons from a Sensor Network Expedition," in *European Workshop on Wireless Sensor Networks (EWSN '04)*, Berlin, Germany, 2004, pp. 66–80.
- [5] G. J. Minden *et al.*, "Architecture and Prototype of an Ambient Computational Environment: Final Report," Univ. of Kansas, Tech. Rep. ITTC-FY2004-TR-23150-09, July 2003, http://www.ittc.ku.edu/publications/documents/Minden2003_23150-09.pdf.
- [6] J. Mauro, "Security Model in the Ambient Computational Environment," Master's thesis, Dept. of EECS, The University of Kansas, USA, 2004, http://www.ittc.ku.edu/research/thesis/documents/james_mauro_thesis.pdf.
- [7] M. Botts *et al.*, "OGC Sensor Web Enablement: Overview and High Level Architecture, OGC 06-050r2," <http://www.opengeospatial.org/pt/06-046r2>.
- [8] B. L. Gorman *et al.*, "Advancing Sensor Web Interoperability," *Sensors*, vol. 22, no. 4, pp. 14–18, April 2005, <http://www.sensorsmag.com/sensors/Homeland+Security/Advancing-Sensor-Web-Interoperability/ArticleStandard/Article/detail/185897>.
- [9] P. B. Gibbons *et al.*, "IrisNet: An Architecture for a Worldwide Sensor Web," *IEEE Pervasive Computing*, pp. 22–33, Oct-Dec 2003.
- [10] "The IEEE 1451 Standard," <http://ieee1451.nist.gov/>.
- [11] M. Botts, "Technical Specification for Sensor Model Language (SensorML) - Version 0.0, Open Geospatial Consortium, OGC 05-086r2," http://portal.opengeospatial.org/files/?artifact_id=13879.
- [12] D. T. Fokum *et al.*, "A Taxonomy of Sensor Network Architectures," University of Kansas, Tech. Rep. ITTC-FY2008-TR-41420-07, January 2008.
- [13] M. Blaze *et al.*, "The KeyNote Trust-Management System Version 2," RFC 2704, September 1999.
- [14] S. Muralidharan *et al.*, "SensorNet Architecture with Multiple Owners," University of Kansas, Tech. Rep. ITTC-FY2008-TR-41420-02, July 2007.
- [15] "Cyanose 320 Handheld Electronic Nose," <http://www.smithsdetection.com/eng/1383.php>.
- [16] "Microsoft's Universal Plug and Play (UPnP)," <http://technet.microsoft.com/en-us/library/bb457049.aspx>.
- [17] "Optical Sensor Interface Standard," http://www.ntb.ch/pub/bscw.cgi/d18647/OSIS_WG2_Standard_Documentation.pdf.

- [18] “*The IEEE 1451.0 Standard: A Smart Transducer Interface for Sensors and Actuators - Common Functions, Communications Protocols and Transducer Electronic Data Sheets (TEDS) Formats,*” <http://grouper.ieee.org/groups/1451/0/>.
- [19] “*Java Distributed Data Acquisition and Control (JDDAC),*” <https://jddac.dev.java.net/>.
- [20] “*MICA MOTES from Crossbow Technologies,*” <http://www.xbow.com/Products/productdetails.aspx?sid=164>.
- [21] “*TinyOS,*” <http://www.tinyos.net/>.
- [22] “*TinyAODV,*” <http://tinyos.cvs.sourceforge.net/tinyos/tinyos-1.x/contrib/hsn/>.
- [23] “*Mobile Ad hoc Networking (MANET) Charter,*” <http://www.ietf.org/html.charters/manet-charter.html>.
- [24] C. E. Perkins *et al.*, “Ad hoc On-demand Distance Vector (AODV) Routing,” IETF RFC 3561.
- [25] Sun Microsystems, “Java Remote Method Invocation (Java RMI),” <http://java.sun.com/products/jdk/rmi/>.
- [26] “*Sun Small Programmable Object Technology (SunSpots),*” <http://www.sunspotworld.com/>.
- [27] “*GumStix,*” <http://gumstix.com/>.

The University of Kansas



Technical Report

**Modeling for Analysis and Design of
Communications Systems and Networks for
Monitoring Cargo in Motion Along Trusted Corridors**

Daniel T. Fokum and Victor S. Frost

ITTC-FY2010-TR-41420-25

May 2010

Project Sponsor:
Oak Ridge National Laboratory
Award Number 4000043403

Copyright © 2010:
The University of Kansas
2335 Irving Hill Road, Lawrence, KS 66045-7559
All rights reserved.

Abstract

Exports from Asia to the United States have increased significantly in recent years, causing congestion at ports on the Pacific coast of the United States. To alleviate this congestion, some groups want to ship goods by rail directly from ports to inland intermodal traffic terminals. However, for such an effort to succeed, shippers must have “visibility” into the rail shipment. In this research we seek to provide visibility into shipments through optimal placement of sensors and network elements. We formally define the notion of visibility and then develop models to identify and locate network elements and containers on trains. Two models have been developed—one for use when all network elements are on the train and the other for use when some are located trackside—to determine sensor placements and network design. The models show that, under reasonable assumptions, sensor deployment reduces the overall system cost; therefore, sensor networks make sense for monitoring cargo. These models also enable the study of system trade-offs while achieving the desired level of visibility into cargo shipments.

CONTENTS

I	Introduction	1
I-A	Visibility	1
I-B	Problem Statement	2
I-C	Metrics	3
II	A System Description for Identifying and Locating System Elements	4
II-A	Identification	4
II-B	Location	4
III	Parameters and Variables	5
III-A	Parameters	6
III-A1	Container Assignment Parameters	6
III-A2	Communications Systems Assignment Parameters	6
III-A3	Distributions for Decision Maker Notification	7
III-B	Communications Systems Assignment Variables	12
III-B1	Train-Mounted Deployment Variables	12
III-B2	Trackside Deployment Variables	14
IV	Model Descriptions	15
IV-A	Train-mounted Deployment	16
IV-A1	Objective Function	16
IV-A2	Constraints	16
IV-B	Trackside Deployment with Fixed Train Speeds	17
IV-B1	Objective Function	18
IV-B2	Constraints	18
IV-C	Trackside Deployment with Variable Train Speeds	18
IV-D	Extending the Sensor Placement Models	19
IV-E	Container Placement	23
V	Model Growth and Validation	23
V-A	Model Growth and Computational Complexity	23
V-B	Model Validation	24

V-B1	Train-Mounted Model	24
V-B2	Trackside Model with Fixed Speeds	25
VI	Related Work	27
VI-A	Mixed Integer Linear Programs	28
VI-B	Mixed Integer Nonlinear Programs	28
VII	Conclusion	29
	Acknowledgments	29
	References	30

LIST OF FIGURES

1	Unit with Two 20 ft. Containers and One 40 ft. Container	5
2	Two Well-cars with Load Indices Identified	6
3	Container Seal	9
4	Sequence Diagram with Messages Involved in Decision Maker Notification	10
5	Example Train With Sensors Assigned	20
6	Problem Growth in Number of Variables and Constraints	24
7	Train-mounted Model: Sensor Locations and Cost Metric Variation with Number of Visible Containers	25
8	Train-mounted Model: Trip Duration and Pr[Event Occurrence] versus Cost Metric	26
9	Trackside Model: Reporting Deadline versus Reader Separation and Cost Metric	26
10	Trackside Model: Train Speed versus Cost Metric and Sensor Transmission Range	27

LIST OF TABLES

I	Train-related Parameters	8
II	Sensor and Communications Equipment-related Parameters	8
III	Message-related Parameters	8
IV	Communications System Probability Parameters	9
V	Cost Parameters	9
VI	Statistics for Time Taken in Seconds Between Seal Events and Decision Maker Notification for Short-haul Trial and Empirical Data	11

VII	Estimated Gamma Distribution Parameters for Time Taken Between Seal Events and Decision Maker Notification	11
VIII	Train-Mounted Deployment Variables	12
IX	Trackside Deployment Variables	14
X	Parameters used in Validating Models	21
XI	Additional Parameters used in Validating Models	25

I. INTRODUCTION

IN recent years exports from Asia to the USA have increased significantly, resulting in bottlenecks at certain key ports on the West Coast. Some groups involved in freight transportation have sought to get around the bottlenecks at Pacific Coast ports by using inland ports. To this end, they seek to offload cargo from ships directly onto trains destined for an inland intermodal traffic terminal. Once at the terminal, the freight can be processed by Customs and then distributed within the United States.

For the success of the scheme described above, shippers need to gain “visibility” into freight and cargo movement, particularly in intermodal “black holes,” where freight changes hands across modes and carriers. Visibility will only be possible through real-time integration of sensor data with carrier, shipper, broker, importer, exporter, and forwarder information. Unfortunately, different complex systems are currently used in the container transport chain [1].

To achieve the objective of providing visibility into cargo shipments, trains, railcars, and containers will be equipped with sensors and devices that communicate sensor status, sensor ID, and train location. Breaking a sensor on a container would generate a signal that is communicated to a reader over a network and then to train personnel and/or to an operations center as an alarm message in near real-time. In addition, location information will be sent with the alarm so that the geographic location of the breakage event can be identified. Shipment information from a Trade Data Exchange (TDE) [2] will be included in the alarm so that the the rail car, container, and its contents can be identified. While sensors will present a non-negligible initial cost, their use could allow the sensing system to demonstrate shipment integrity. It is also expected that the use of such systems may help reduce the risk of cargo theft, which the Federal Bureau of Investigation (FBI) estimates costs the U.S. economy \$15–\$30 billion dollars every year [3].

The objective of the research presented here is to develop models to find the “best” system design including communications network and locations for sensors in a rail-based sensor network, as well as to guide the design of future cargo monitoring systems. These models can also be applied to determine system trade-offs when monitoring cargo in motion.

A. *Visibility*

In this section we provide a formal definition of visibility. Events are recorded in the cargo monitoring sensor network whenever an attempt is made to open, close, or tamper with a seal. The seal also generates other events during normal operations. Informally the integrity of a cargo shipment has state. These states will include locking the container and closing the seal, opening the seal and then the container, and

$$\nu(j, t, \tau, \text{TR}_j, P_\epsilon, E_j, P_\alpha, F_j) = \begin{cases} 1 & \text{if } (\Pr(t \leq \tau) \geq \text{TR}_j \text{ AND } P_\epsilon \geq E_j \text{ AND } P_\alpha \leq F_j) \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

tampering with the seal. A critical event is generated whenever the integrity of a shipment changes states. Examples of critical events include messages indicating that seals are opened, closed, or tampered with. Messages are also generated during normal operation of the cargo monitoring system. These messages denote maintenance events and examples include alerts indicating an armed seal or a seal warning of a low battery. The set of messages will also include items such as a seal incorrectly reporting a tamper event, incorrectly being detected as missing, or an incorrect low battery report. This latter group of events will be considered false alarms. Important aspects of visibility include the likelihood of a sensor detecting an event at a container, the time taken by a sensor to notify decision makers of an event, and the likelihood of a false alarm from a sensor. We define visibility as a binary variable that relates the probability of detecting an event at a container with the time taken to report that event to the decision maker and the probability of false alarm for that container. More formally we define a container j , as visible if $\nu(j, t, \tau, \text{TR}_j, P_\epsilon, E_j, P_\alpha, F_j) = 1$, where the visibility function is defined as in equation (1) and the parameters of the function are:

- An event can be detected at container j , and made known to the decision maker with a probability P_ϵ , that is greater than or equal to some threshold, E_j .
- The time t , taken to notify the decision maker of an event, must lie within an interval of length τ , with probability greater than or equal to some threshold TR_j , i.e., $\Pr(t \leq \tau) \geq \text{TR}_j$.
- The probability of false alarm at container j , P_α , must be kept less than or equal to some threshold F_j .

The system design determines P_ϵ , P_α , and $\Pr(t < \tau)$. The combination of P_ϵ , P_α and $\Pr(t < \tau)$ can be mapped into a visibility space.

B. Problem Statement

In this section we introduce a generalized problem statement. We may be able to achieve visibility into a cargo shipment on a train by placing sensors, readers, and backhaul communication devices on every container on a train (as is done today for high-value cargo, e.g., hazardous material), or by deploying sensors on every container on the train and closely placing readers with backhaul communications

capabilities at the trackside. However, the cost and system trade-offs for such approaches are unknown. As a result this research is aimed at answering the following system design question:

Given a collection of containers and a collection of end-to-end information subsystems (including sensors, seals, readers, and networks); how do we design an end-to-end system that provides “visibility” (meeting given E_j , F_j , and TR_j constraints for all containers) while minimizing overall system cost?

In our specific rail scenario, our overall design question spawns the following issues:

- 1) How to map and analyze a “system” description of containers on railcars, train scenario—including train speed and trips per time unit—and associated communications infrastructure into the visibility space? Thus an appropriate system model needs to be developed.
- 2) How to assign a cost to every position in the visibility space?
- 3) How to use 1) and 2) to find minimum “cost” systems for providing visibility into a rail shipment?
- 4) How to use 1) and 2) to determine system trade-offs when seeking visibility into rail shipments?

C. Metrics

Metrics are needed to compare the “goodness” of two or more proposed system designs. In this section we present our metrics, which include:

- **System operational cost.** This metric is computed per trip, and it consists of each sensor’s false alarm cost, the cost of deploying the sensors, repeaters and readers, network, and the backhaul communications devices, as well as the cost of reporting events. The costs of missing an event at a given container as well as the costs of a communications failure at a sensor are also components of this metric.
- **Visibility metric.** We declare that container j is visible if the sensor placed on j meets all the system designer-imposed constraints for visibility of the container.

The rest of this paper is laid out as follows: in Section II we present a scheme for identifying containers, sensors, and the locations that they occupy on trains. Section III introduces the parameters and variables in our model for analysis and design of communications systems for cargo monitoring. We describe our models for optimal sensor and communications system assignment in Section IV. Section V presents arguments for validating our models as well as discussing model growth. In Section VI we discuss related work. Concluding remarks are provided in Section VII.

II. A SYSTEM DESCRIPTION FOR IDENTIFYING AND LOCATING SYSTEM ELEMENTS

Notation is needed to identify sensors and containers in models to analyze and design communications systems and networks for monitoring cargo in motion along trusted corridors. In this section we present one scheme for identifying containers and the locations that they occupy on a train. We focus first on container identification and then turn our attention to indexing container and sensor locations on trains.

A. Identification

The containers that are to be placed on the train typically come in a variety of lengths ranging between 20 ft. (6.1 m) and 53 ft. (16.2 m). We propose sorting the loads by length and numbering each container with a unique index j , that is sufficient for accessing the container's properties, such as its length, weight, value, and other intrinsic attributes of the container that might be necessary to solve the sensor placement problem and identify system trade-offs. For example, suppose we have two 20 ft. (6.1 m) containers, two 40 ft. (12.2 m) containers, and one 45 ft. (13.7 m) container, then j ranges from $1 \dots 5$, with each container bearing a unique j index. The container types can then be identified by using a function that returns the length of a container when given an integer j , i.e., the container lengths could be stored in a vector L , so that L_j indicates the length of container j .

Every sensor, repeater, and backhaul communications device that is to be placed on the containers is identified with a unique index, i . This index, which starts off with value 1, is sufficient for reading the parameters, e.g, transmission range, associated with each communications element.

B. Location

Each railcar consists of one or more permanently attached units, where a unit is a frame that can support one or more slots [4]. Each unit is uniquely identified by an integer k , where k starts off with value 1. The index $k = 0$ is reserved for the locomotive.¹

Review of the Association of American Railroads Loading Capabilities Guide [5] indicates that railcars used for intermodal transportation have at most two positions (layers) for carrying intermodal loads. Within each position, slots are available for holding containers. For example, [5] indicates that two 20 ft. containers can be placed in the bottom position and a 40 ft. container is placed over both 20 ft. containers in the top position, as shown in Fig. 1.

¹The issue of having to deal with several locomotives on a train is not part of this model. If more than one locomotive is present, all the locomotives are treated as one with respect to the goals of this system.

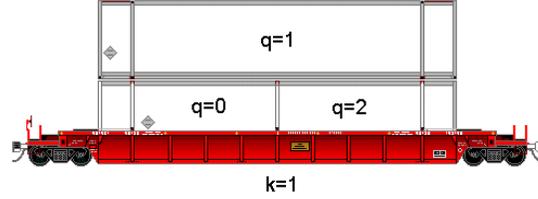


Fig. 1. Unit with Two 20 ft. Containers and One 40 ft. Container

In general, the first available slot in a unit, i.e., the first slot in the bottom position is marked with index $q = 0$. All other slots in the bottom position have even q indices. The first slot in the top position is always indexed with index $q = 1$, while all other slots in the top position will have odd indices.²

From above we see that the integer triple (j, q, k) is sufficient for identifying the unit and slot occupied by container j . For instance, using the five container example presented in Section II-A, the integer triple $(1, 0, 2)$ implies that container 1 is found in slot 0 of unit 2. Similarly, $(5, 1, 1)$ implies that container 5 is found in slot 1 of unit 1.

In this section we presented an orthogonal indexing system for containers on a train. This numbering scheme is based on assigning containers with a unique integer j , that is used to identify containers and to retrieve additional container attributes, an index k , that is used to identify railcars, and an integer q , used to identify slots on a railcar.

III. PARAMETERS AND VARIABLES

In this section we use the container identification and location scheme from Section II to introduce the parameters and variables in two models for computing the system cost metric for a cargo monitoring system. Parameters will be given to the system designer for a specific placement problem, while the optimization process will assign appropriate values to the variables such that the objective is attained while satisfying any design constraints. To facilitate the presentation of the variables and parameters, throughout this section we use the five container example shown in Fig. 2. The rest of this section is laid out as follows: Section III-A introduces the parameters for the models. For the sake of completeness there is a very brief discussion on container assignment parameters in Section III-A1 while the communications system parameters are presented in Section III-A2. Section III-A3 uses probability distributions to deter-

²A review of the Association of American Railways Loading Guide [5] indicates that we will rarely have more than one container in the top position.

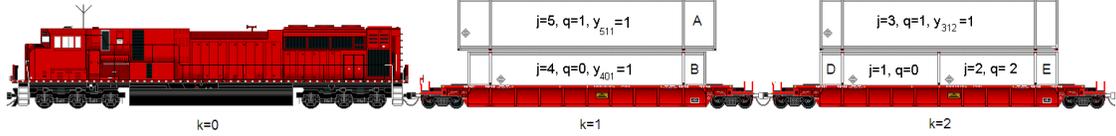


Fig. 2. Two Well-cars with Load Indices Identified

mine the likelihood of timely decision maker notification. The variables for the models are introduced in Section III-B.

Suppose that we have a train with a locomotive and two well-cars as shown in Figure 2. Furthermore, assume that we have two 20 ft. containers, two 40 ft. containers, and one 45 ft. container. Recall that each container (load) is uniquely identified by an integer j , while the railcars are uniquely identified by an index k . Assume that the containers are indexed³ such that $j = 1$ refers to the most expensive 20 ft. container, while $j = 2$ refers to the least expensive 20 ft. container, $j = 3$ and $j = 4$ refer to the 40 ft. containers. Finally, $j = 5$ is used to denote the 45 ft. container.

A. Parameters

This section introduces the parameters for models. First, we discuss the container assignment parameters, which indicate valid container assignments as well as information on container and railcar attributes. Next, we discuss the communications systems assignment parameters. Finally, we present some distributions to model the time taken to notify decision makers of events on a train.

1) *Container Assignment Parameters*: This section introduces the container assignment parameters for our model. The length of the k^{th} unit is represented by U_k and the length of the j^{th} container is given by L_j . The binary parameter y_{jqk} indicates a given container's location on the train, and it is defined as:

$$y_{jqk} = \begin{cases} 1 & \text{if } j^{\text{th}} \text{ container is assigned to slot } q \text{ in unit } k, \\ 0 & \text{otherwise} \end{cases}$$

2) *Communications Systems Assignment Parameters*: In this section we introduce parameters that are necessary to address the system design, including the sensor assignment portion of the container assignment and sensor assignment problem.

³Note that we are not required to index the containers by value. The indices could be randomly assigned as long as each number is used exactly once.

Suppose each of the containers has a value v_j . Furthermore, suppose that each time the decision maker receives notification of an event at a container within a time interval τ_j we get a savings σ_j (Note that σ_j can be greater than v_j). These savings can be viewed as the value of a detected event to the decision maker. We assume that all the repeaters and backhaul communications devices on the train are arranged in a linear topology.

Backhaul communications devices in our system are used to transmit event reports from the train to the decision maker (possibly via an operations center). We use the binary parameter B_{qk} to indicate when a backhaul communications device is placed in slot q of unit k . This parameter is defined as:

$$B_{qk} = \begin{cases} 1 & \text{if backhaul communications device is placed} \\ & \text{in slot } q \text{ of unit } k, \\ 0 & \text{otherwise} \end{cases}$$

In our system, sensors have a limited transmission range, and they are interrogated by more powerful radios called “repeaters/readers.” The repeaters can communicate with each other over longer distances to get event reports to a backhaul communications device. We use the binary parameter A_{qk} to indicate when a repeater is placed in slot q of unit k . This parameter is defined as:

$$A_{qk} = \begin{cases} 1 & \text{if repeater is placed in slot } q \text{ of unit } k, \\ 0 & \text{otherwise} \end{cases}$$

There are other communications systems assignment parameters used in the optimal placement model. These parameters are shown as follows: Table I presents train-related parameters, sensor and communications equipment-related parameters are listed in Table II, Table III presents message-related parameters, communications system probability parameters are defined in Table IV, and all the cost parameters in the model are listed in Table V.

3) *Distributions for Decision Maker Notification:* In parallel with the modeling work being described here we have also built a Transportation Security Sensor Network (TSSN) for monitoring rail cargo in motion [6]. In the context of this work we envision that sensors will be placed on intermodal containers that are being shipped by rail, as shown in Fig. 3.

The sensors on the containers will allow the TSSN to detect events at shipping containers and report those that are important to decision makers using commercial networks. Experiments have been carried out with the prototype TSSN and empirical data has been collected [6], [7]. The empirical data will be used to enhance the models described here.

TABLE I
TRAIN-RELATED PARAMETERS

Parameter	Comment
D	Rail trip duration in hours.
d_T	Length of rail journey in kilometers.
\dot{x}	Train-speed in kilometers per hour.
t_f	Number of trains passing a given trackside reader per hour.
t_L	Number of trips per locomotive per hour.
ζ	Probability of event occurrence during a trip.

TABLE II
SENSOR AND COMMUNICATIONS EQUIPMENT-RELATED PARAMETERS

Parameter	Comment
LT_A	Useful lifetime of trackside reader in hours.
LT_c	Useful lifetime of cellular communications device in hours.
LT_s	Useful lifetime of satellite communications device in hours.
FP_2	Weight of sensor cost allocated to improving event detection.
FP_3	Weight of sensor cost allocated to improving timely reporting or successful communications in train-mounted and trackside cases, respectively.
FP_4	Weight of sensor cost allocated to reducing false alarms.
FP_5	Weight of sensor cost allocated to improving sensor transmission range.
FP_6	Weight of sensor cost allocated to reducing sensor read time.

TABLE III
MESSAGE-RELATED PARAMETERS

Parameter	Comment
l	Average message length in bytes between sensor and operations center.
λ_i	Message generation rate for sensor i .
RTT_c	Communications round trip time in seconds from train to operations center over the cellular link.
RTT_s	Communications round trip time in seconds from train to operations center over the satellite link.

One of the critical metrics for TSSN performance—and also for visibility—is the time between event occurrence and decision maker notification. Due to the proposed definition for visibility, we need to

TABLE IV
COMMUNICATIONS SYSTEM PROBABILITY PARAMETERS

Parameter	Comment
$Pr(H)$	Probability of successfully transmitting a message from the train to the operations center over the cellular link.
$Pr(I)$	Probability of successfully transmitting a message from the train to the operations center over the satellite link.

TABLE V
COST PARAMETERS

Parameter	Comment
C_{α}	Cost of one false alarm.
C_s	Cost of sending one byte by satellite.
C_c	Cost of sending one byte by cellular.
C_A	Acquisition cost of one reader/repeater.
C_F	Fixed cost of acquiring a sensor.
C_{BC}	Acquisition cost of one backhaul communications device (cellular).
C_{BS}	Acquisition cost of one backhaul communications device (satellite).
C_{HL}	Installation cost of one sensor/seal.
C_{AL}	Installation cost of one reader/repeater.
C_{AD}	Installation cost of one trackside reader.
C_{BD}	Installation cost of one trackside cellular communications device.



Fig. 3. Container Seal

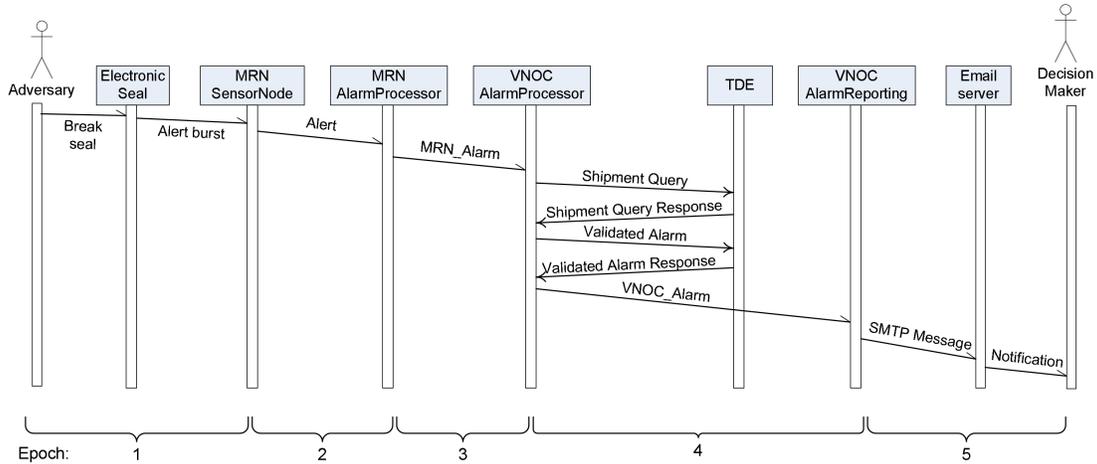


Fig. 4. Sequence Diagram with Messages Involved in Decision Maker Notification

create models that can predict the probability that a decision maker will be notified within a specified time interval. Please refer to Fig. 4 for a sequence diagram showing the messages that are exchanged within the TSSN to notify a decision maker.

In Fig. 4 the electronic seal, Mobile Rail Network (MRN) SensorNode, and MRN AlarmProcessor are on the train. The VNOc AlarmProcessor and Virtual Network Operating Center (VNOc) AlarmReporting services run on a server off the train. Finally, the Trade Data Exchange (TDE) is outside the shipper's network. As shown in Fig. 4 there are five epochs between an event taking place and decision maker notification on a mobile phone.

We would like to generate a distribution that measures the likelihood of timely decision maker notification of an event occurrence. It is reasonable to assume statistical independence of the epochs shown in Fig. 4 because the time taken to break a seal and generate an alert message is independent of the time taken to transfer a message from the MRN to the VNOc. Thus, the probability distribution of the time from event occurrence to decision maker notification is the convolution of the probability distributions for the five epochs listed above.

Based on our experiments we have calculated means and variances for the time taken to transmit a message in each of the epochs listed above. These statistics are summarized in Table VI.

The time epochs shown in Fig. 4 are random. We assume that these random variables can be modeled

TABLE VI
STATISTICS FOR TIME TAKEN IN SECONDS BETWEEN SEAL EVENTS AND DECISION MAKER NOTIFICATION FOR
SHORT-HAUL TRIAL AND EMPIRICAL DATA

Epoch	Description	Min.	Max.	Mean	Median	Std. Dev.
1	Event occurrence to alert generation	0.81	8.75	2.70	2.13	1.86
2	Alert generation to MRN AlarmProcessor Service	0.01	0.08	0.02	0.01	0.01
3	One-way delay from MRN AlarmProcessor to VNOC AlarmProcessor	0.45	2.90	1.89	1.94	0.62
4	MRN_Alarm arrival at VNOC to AlarmReporting Service	0.01	3.01	0.17	0.05	0.32
5	Elapsed time from VNOC AlarmReporting Service to mobile phone	5.2	58.7	11.9	9.8	7.4

TABLE VII
ESTIMATED GAMMA DISTRIBUTION PARAMETERS FOR TIME TAKEN BETWEEN SEAL EVENTS AND DECISION MAKER
NOTIFICATION

Epoch	Description	$\hat{\alpha}$	$\hat{\theta}$
1	Event occurrence to alert generation	4.01	0.60
2 + 4	Alert generation to MRN AlarmProcessor Service and MRN_Alarm arrival at VNOC to AlarmReporting Service	1.13	0.13
3	One-way delay from MRN AlarmProcessor to VNOC Alarm-Processor	13.95	0.14
5	Elapsed time from VNOC AlarmReporting Service to mobile phone	10.44	1.00

using Gamma⁴ probability density functions. The parameters for the distributions are estimated from the collected data and shown in Table VII, where $\hat{\alpha}$ and $\hat{\theta}$ represent the shape and scale parameters respectively. Using the results from [8] we see that there is 99.9 % chance that a decision maker is notified of an event within 4 minutes.

⁴This distribution assumption is based on the positive values and the asymmetric histograms of the observed data. However, this assumption is based on a limited number of samples and additional experiments are needed to validate this claim.

TABLE VIII
TRAIN-MOUNTED DEPLOYMENT VARIABLES

Variable	Comment
α	Probability of false alarm for sensor.
ϵ	Probability of event detection by sensor.
φ	Probability of successful end-to-end communications from sensor to operations center.
C_H	Acquisition cost of one sensor/seal.
Γ_k	Cost of false alarms per railcar.
Δ_k	Cost of missed detection per railcar.
Ξ_k	Cost of reporting an event outside of desired deadline for container visibility.
Λ_k	Cost of transmitting messages generated by all the sensors on a railcar.
Ψ_k	Cost of acquiring and installing sensors on each railcar.
Υ_k	Cost of acquiring and installing repeaters on each railcar.
Ω_k	Cost of acquiring and installing one backhaul communications device on each railcar.

B. Communications Systems Assignment Variables

This section presents the variables used to indicate communications system assignment in our models. These variables are either integers or positive real numbers. Appropriate values will be assigned to these variables such that the best objective function value is attained. First, we present the variable that is common to the trackside and train-mounted cases. Next, we present the other variables that are unique to each case. In general, whenever a variable or parameter is indexed by $q = 0$, and $k = 0$ it is assumed that we will be referring to the locomotive. For example, $A_{00} = 1$ and $B_{00} = 1$ will indicate that a reader and a backhaul communications device, respectively, are located on the locomotive. In our discussion, a “sensor” refers to the combination of sensing and communication devices, e.g., the seal shown in Fig. 3. The binary variable S_{ijqk} indicates when sensor i is assigned to the j^{th} container. This variable is defined as:

$$S_{ijqk} = \begin{cases} 1 & \text{if sensor } i \text{ is attached to } j^{\text{th}} \text{ container in slot } q \text{ of unit } k, \\ 0 & \text{otherwise} \end{cases}$$

1) *Train-Mounted Deployment Variables*: There are other variables, in addition to S_{ijqk} , used for the case when the sensors and related communications infrastructure are on the train. Table VIII presents these variables and equations (2)–(9) show how the variables are computed.

$$\Gamma_k = C_\alpha \sum_{\forall i,j,q} \alpha S_{ijqk} y_{jqk} \quad (2)$$

$$\Delta_k = \zeta \left(\sum_{\forall j,q} \sigma_j y_{jqk} - \sum_{\forall i,j,q} \epsilon \sigma_j S_{ijqk} y_{jqk} \right) \quad (3)$$

$$\Xi_k = \zeta \left(\sum_{\forall j,q} \sigma_j y_{jqk} - \sum_{\forall i,j,q} \varphi \sigma_j S_{ijqk} y_{jqk} \right) \quad (4)$$

$$\Lambda_k = D(\Pr(H)C_c + \Pr(I)(1 - \Pr(H))C_s)l \sum_{\forall i,j,q} \lambda_i S_{ijqk} y_{jqk} \quad (5)$$

$$C_H = C_F + \epsilon \text{FP}_2 + \varphi \text{FP}_3 + (1 - \alpha) \text{FP}_4 \quad (6)$$

$$\Psi_k = \sum_{\forall i,j,q} (C_H + C_{HL}) S_{ijqk} y_{jqk} \quad (7)$$

$$\Upsilon_k = \sum_{\forall q} (C_A + C_{AL}) A_{qk} \quad (8)$$

$$\Omega_k = \left(\frac{C_{\text{BC}}}{t_L \times \text{LT}_c} + \frac{C_{\text{BS}}}{t_L \times \text{LT}_s} \right) \sum_{\forall q} B_{qk} \quad (9)$$

The cost of false alarms per rail car is given by equation (2). This is given by the cost of each false alarm times the sum of probabilities of false alarm for all the sensors that are currently used. Assume that if an event is detected and reported in a timely manner, then there is no loss to the decision maker. In addition, assume that the probability of an event occurring at a container is independent of the probability of a sensor detecting that event or reporting it in a timely manner. Equation (3) computes the cost of a missed detection per railcar, which is given by the probability of event occurrence times the savings that are lost if an event is not detected. Similarly, equation (4) computes the cost of reporting an event outside the required deadline for container visibility. This cost is given by the probability of event occurrence times the savings that are lost if the event is not reported in a timely manner. Equation (5) computes the cost of transmitting messages generated by all the sensors on a railcar. This cost is given by the rail trip duration times the mean cost of transmitting one byte times the sum of message generation rates for all sensors in use. The unit cost of acquiring a sensor for the train-mounted deployment is captured in equation (6). This cost is given by adding up the fixed cost of acquiring each sensor, plus the cost of getting a sensor with specified probabilities of detection, timely reporting, and false alarm. The cost of acquiring and installing the sensors on a railcar is given by substituting equation (6) into (7). Repeater acquisition and installation costs per unit are computed with equation (8). Finally, equation (9) calculates the cost of acquiring and installing a backhaul device on each rail car. We assume that the backhaul

TABLE IX
TRACKSIDE DEPLOYMENT VARIABLES

Variable	Comment
α	Probability of false alarm for sensor.
ϵ	Probability of event detection by sensor.
ρ	Probability of successful communications between trackside reader and sensor.
β	Rate of change of probability of unsuccessful communications with train speed.
η	Probability of unsuccessful communications between trackside reader and sensor when both are stationary.
θ	Real number that specifies the minimum sensor transmission range in meters.
t_{Read}	Real number that states the maximum time in seconds available to read the sensors.
C_H	Acquisition cost of one sensor/seal.
Γ_k	Cost of false alarms per railcar.
Δ_k	Cost of missed detection per railcar.
Ξ_k	Cost of unsuccessful communications between a trackside reader and the sensors on a railcar.
Λ_k	Cost for transmitting messages generated by all the sensors on a rail car.
Ψ_k	Cost of acquiring and installing sensors on each railcar.

devices are reused for several trips, thus we amortize this cost over the expected number of trips in the device's lifetime.

2) *Trackside Deployment Variables*: The variable S_{ijqk} , which is defined above, is also used when the sensors are mounted on the train and the readers are at the trackside. The rest of the variables for the trackside deployment case are defined in Table IX and equations (2), (3), (7), and (10)–(13) show how the variables are computed.

$$\rho = 1 - (\eta + \dot{x}\beta) \quad (10)$$

$$C_H = C_F + \epsilon\text{FP}_2 + \rho\text{FP}_3 + (1 - \alpha)\text{FP}_4 + \theta\text{FP}_5 + \frac{\text{FP}_6}{t_{\text{Read}}} \quad (11)$$

$$\Xi_k = \zeta \left(\sum_{\forall j,q} \sigma_j y_{jqk} - \sum_{\forall i,j,q} \rho \sigma_j S_{ijqk} y_{jqk} \right) \quad (12)$$

$$\Lambda_k = \left(\frac{d_T}{\dot{x}} \right) C_{cl} \sum_{\forall i,j,q} \lambda_i S_{ijqk} y_{jqk} \quad (13)$$

Suppose that we are given that the probability, ρ , of successful communications from a sensor to a reader varies with train speed according to equation (10). In the trackside case the optimization process will determine appropriate values for α , ϵ , ρ (including η and β), θ , and t_{Read} . The cost of acquiring one sensor for the trackside case is given by equation (11). The cost, Ψ_k , of acquiring and installing sensors on each railcar in the trackside case is given by substituting equation (11) into (7). The values for the Γ_k and Δ_k variables are computed using equations (2) and (3), respectively. As we did above we assume that the likelihood of an event occurring at a container is independent of a sensor detecting that event or the timely notification of that event. In this case we assume that events will get to the operations center in a timely manner if the sensors are read by a trackside reader. The cost of trackside reader failing to read a sensor is given by equation (12). This cost is given by the probability of an event times the cost of a trackside reader failing to read a sensor. Equation (13) computes the cost of transmitting all the messages generated by all the sensors on a railcar. This cost is given by the rail trip duration times the cost of transmitting one message times the message generation rates for all the sensors on a railcar.

IV. MODEL DESCRIPTIONS

In this section we present two models for computing the cost metric for a system that uses sensors for cargo monitoring. The models that we develop here are robust enough to handle the following sensor deployment cases:

- A deployment of sensors and a backhaul communications device on the train. This case can be further divided into two subcases:
 - The sensors cannot engage in multihop communications. Instead, they can only communicate with the repeaters or the backhaul communications device. We call this the hierarchical deployment case.
 - The sensors can engage in multihop communications to forward messages to the backhaul communications device. As a result, this case does not contain any dedicated repeaters. We call this the ad hoc deployment case.
- A deployment of sensors to the train, while the readers and backhaul communications devices are at the trackside. This case can also be split into two subcases for when the train speed is fixed and when it is allowed to vary.

The first model, which is presented in Section IV-A, is used when the backhaul communication devices and repeaters are placed on a train. Section IV-B presents the second model, which is used when the train's speed is fixed and backhaul communication devices and readers are placed trackside. Section IV-C shows

how the trackside model can be applied in the case where the train speed is allowed to vary. The models discussed in this section are presented using the following general optimization problem formulation:

$$\begin{aligned} & \text{minimize } f_o(x;p) \\ & \text{subject to } f_i(x;p) \leq b_i, \quad i = 1, \dots, m \end{aligned}$$

The objective function, $f_o(x;p)$, will be the system cost metric function, which depends on a vector of variables, x , and a vector of parameters, p . The constraints of the optimization problem are defined by the m f_i equations. When necessary we provide comments relevant to the equations inline.

In Section IV-D we show how the hierarchical sensor deployment can be mapped to the ad hoc sensor deployment case. Our analysis in the next four subsections assumes that the containers on the train are already placed in fixed locations on the train. Thus, Section IV-E briefly mentions an optimization-based approach which can be used to place containers on trains.

A. Train-mounted Deployment

In this subsection we present a model to minimize the system cost metric of a cargo monitoring system when the sensors and backhaul communications device are on the train. First, we present the objective function and then we discuss the model's constraints, which define valid container and sensor placements.

1) *Objective Function:* Equation (14) computes the system cost metric over the duration of a trip:

minimize

$$\sum_k (\Gamma_k + \Delta_k + \Xi_k + \Lambda_k + \Psi_k + \Upsilon_k + \Omega_k) \quad (14)$$

The objective function sums the cost of false alarms over a rail journey, cost of missing a detection at a given container, the cost of a sensor failing to communicate in a timely manner, the cost of communications across a rail journey, the material and installation costs of sensors and repeaters, respectively. Finally, the last term in the sum computes the material and installation cost of the backhaul communications device.

2) *Constraints:* The following constraints must be valid for any given optimal deployment of sensors to containers on a train.

subject to

$$\sum_{\forall j,q,k} S_{ijqk} \leq 1 \quad \forall i \quad (15)$$

$$\sum_{\forall i,q,k} S_{ijqk} \leq 1 \quad \forall j \quad (16)$$

Certain attributes (for example, transmission range, detection probability, and false alarm rate) of the sensors, repeaters, and backhaul communications devices are unique to the network elements. Thus, if a given sensor, for example, is placed on a certain container that same sensor cannot be used on another container. Equation (15) ensures that each sensor cannot be simultaneously assigned to more than one container, while equation (16) ensures that each container has no more than one sensor.

$$\varphi = \Pr(t \leq \tau) \quad (17)$$

$$\sum_{\forall i,q,k} \varphi S_{ijqk} y_{jqk} \geq \text{TR}_j \quad \forall j \quad (18)$$

In equation (17) we use the probability distribution defined in Section III-A3 to look up the probability of timely notification. Equation (18) enforces one of the visibility requirements for container j . In (18) we require that t , the time taken by a sensor to notify a decision maker of an event, must lie within an interval τ , with probability exceeding some threshold TR_j .

$$\sum_{\forall i,q,k} \epsilon S_{ijqk} y_{jqk} \geq E_j \quad \forall j \quad (19)$$

Equation (19) requires that events are detected at container j with a probability ϵ , that exceeds some threshold E_j .

$$\sum_{\forall i,q,k} \alpha S_{ijqk} y_{jqk} \leq F_j \quad \forall j \quad (20)$$

Equation (20) enforces the third component of the visibility requirement. In (20) we require that the probability of false alarm at container j , α must be kept lower than some threshold F_j . Equations (18)–(20) ensure that only solutions in the visibility space are considered.

B. Trackside Deployment with Fixed Train Speeds

In this subsection we present a model to minimize the system cost metric of a cargo monitoring system when the backhaul communications devices and readers are trackside. In this case the train's speed is fixed; however, the probability of successful communications from the sensors to the readers varies with train speed. We intend to study the system trade-offs that exist when monitoring rail-borne cargo. This second model facilitates exploration of the trade-off space by capturing the metrics of a different cargo monitoring methodology, which can be compared with the metrics of the first model. As was done above, the objective function is presented first followed by a discussion of the constraints for this model.

1) *Objective Function:* Equation (21) computes the system cost metric for a trackside-based freight monitoring system over the duration of a trip:

minimize

$$\sum_k (\Gamma_k + \Delta_k + \Xi_k + \Lambda_k + \Psi_k) + \left(\left(\frac{C_A + C_{AD}}{t_f \times \text{LT}_A} + \frac{C_{BC} + C_{BD}}{t_f \times \text{LT}_c} \right) \times \left\lfloor \frac{d_T}{d_A} \right\rfloor \right) \quad (21)$$

The sum in the objective function captures the cost of false alarms over a rail journey, the savings that are lost when a sensor either fails to detect that an event has occurred at a container, the cost of communications across a rail journey, and the material and installation costs of sensors. Finally, the last term captures the cost of setting up trackside readers along a given route.

2) *Constraints:* Equation (15) holds in this case because no sensor can be placed simultaneously on more than one container. We also require that each container can have no more than one sensor, thus, equation (16) is also valid in this case. In addition, equations (19) and (20) are visibility requirements, thus they are also applicable in this case. Finally, the following constraints must also apply:

subject to

$$2\theta - \dot{x}t_{\text{Read}} \geq 0 \quad (22)$$

Equation (22) says that the minimum time that a sensor is within range of a trackside reader must be greater than the time taken to read a sensor. This constraint allows the train's speed to be limited such that the trackside reader has enough time to read the sensor.

$$2\theta - \dot{x}_{\text{Max}}t_{\text{Read}} \leq 0 \quad (23)$$

Equation (23) states that sensor view time must be less than or equal to the read time if the train is passing the trackside reader at the maximum speed at which a sensor can be read.

$$d_A \leq 2\dot{x}\rho \frac{(\tau - \text{RTT})}{(2 - \rho)} \quad (24)$$

The trackside readers are spaced according to equation (24) so that the expected time for end-to-end communications from any sensor plus the time taken to cover the distance between trackside readers must be less than the message reporting deadline.

C. Trackside Deployment with Variable Train Speeds

In this subsection we present a model to minimize the system cost metric of a cargo monitoring system when the backhaul communications devices and readers are trackside and the train speed can be varied

based on sensor parameters. In this case we are optimizing over sensor locations, train speed, and reader separations. This change can be accommodated using equation (21) as the objective function.

Constraints: Equations (15), (16), (19), (20), (22), and (24) also hold in this case for the same reasons advanced in Section IV-B2. On the other hand equation (23) does not hold since the train speed is not fixed.

D. Extending the Sensor Placement Models

The presence of repeaters in any system deployment for cargo monitoring adds one more layer of complexity. In Section IV we claimed that a deployment where the sensors can only communicate with repeaters or a backhaul communications device on the train is related to a deployment in which the sensors can engage in multihop communications to forward messages to the backhaul communications device. In this section we discuss how to map the hierarchical deployment case to an ad hoc deployment. In demonstrating this mapping we make the following assumptions:

- The sensor deployment in the hierarchical case is dense enough to have, in the ad hoc case, a fully connected network of sensors with multihop communications capabilities.
- The visibility constraints are the same in all cases and these constraints determine which containers get sensors.
- The probabilities of detection, timely reporting, and false alarm for the sensors do not change as we go from the hierarchical to the ad hoc deployment case.
- Each case contains the same number of backhaul communications devices.
- The ad hoc deployment case does not contain any repeaters.

Suppose that CM_{Hier} and CM_{AD} represent the cost metrics for the hierarchical and ad hoc deployment cases respectively. Observe that no changes need to be made to the objective function because it simply returns a cost metric when presented with sensor and communications infrastructure locations and their characteristics.

Definitions: Let C_H and C_{HL} represent the acquisition and installation costs for the sensors used in the hierarchical case, while C'_H and C'_{HL} represent the acquisition and installation costs for the sensors used in the ad hoc case. Let J_{Hier} and J_{AD} represent the sets of containers assigned sensors in the hierarchical and ad hoc deployment cases respectively. Let I_{Hier} and I_{AD} represent the sets of communications devices (sensors, repeaters, and backhaul communications) which are assigned in the hierarchical and ad hoc deployment cases, respectively. Furthermore, define S_{Hier} and S_{AD} as the set of sensors in the hierarchical and ad hoc deployment cases. B_{Hier} and B_{AD} and R_{Hier} and R_{AD} are the sets of backhaul communications

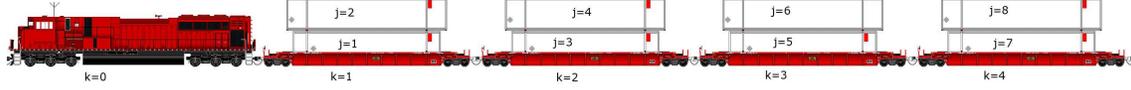


Fig. 5. Example Train With Sensors Assigned

(B_{XX}) and repeaters (R_{XX}) for the hierarchical and ad hoc deployment cases. Then:

$$I_{\text{Hier}} = S_{\text{Hier}} \cup R_{\text{Hier}} \cup B_{\text{Hier}}$$

$$I_{\text{AD}} = S_{\text{AD}} \cup R_{\text{AD}} \cup B_{\text{AD}}$$

We claim that, given the assumptions above, the hierarchical sensor deployment case can be mapped to the ad hoc case. This mapping is based on the assumption that sensors which were previously assigned in the hierarchical deployment case are not moved to other containers in the ad hoc case. This mapping is shown in equation (25).

$$\begin{aligned}
\text{CM}_{\text{AD}} = & \text{CM}_{\text{Hier}} + C_{\alpha} \sum_{\substack{i \in I_{\text{AD}} \setminus I_{\text{Hier}} \\ j \in J_{\text{AD}} \setminus J_{\text{Hier}} \\ \forall q,k}} \alpha S_{ijqk} y_{jqk} - \zeta \sum_{\substack{i \in I_{\text{AD}} \setminus I_{\text{Hier}} \\ j \in J_{\text{AD}} \setminus J_{\text{Hier}} \\ \forall q,k}} \sigma_j S_{ijqk} y_{jqk} (\epsilon + \varphi) \\
& + D(\text{Pr}(H)C_c + \text{Pr}(I)(1 - \text{Pr}(H))C_s)l \sum_{\substack{i \in I_{\text{AD}} \setminus I_{\text{Hier}} \\ j \in J_{\text{AD}} \setminus J_{\text{Hier}} \\ \forall q,k}} \lambda_i S_{ijqk} y_{jqk} \\
& + \sum_{\substack{i \in I_{\text{AD}} \\ j \in J_{\text{AD}} \\ \forall q,k}} (C'_H + C'_{\text{HL}}) S_{ijqk} - \sum_{\substack{i \in I_{\text{Hier}} \\ j \in J_{\text{Hier}} \\ \forall q,k}} (C_H + C_{\text{HL}}) S_{ijqk} - \sum_{\forall q,k} (C_A + C_{\text{AL}}) A_{qk}
\end{aligned} \tag{25}$$

When proving our claim we will use the example train shown in Fig. 5 to illustrate the proof. The train consists of a locomotive and four well cars, with each car bearing two containers. The savings resulting from detecting an event at a container is 8,000 units. The following components are deployed in hierarchical mode for cargo monitoring: a backhaul communications device, a repeater, and seven sensors. The small rectangles on each of the containers in Fig. 5 indicate sensor assignments, while a repeater is on container 6 on railcar 3, and the backhaul communications device is in the locomotive. Finally, assume that we are given the parameter values shown in Table X.

Proof:

- 1) If we map the hierarchical sensor deployment case to the ad hoc deployment case, then we must get rid of any repeaters in the deployment (Recall that the ad hoc deployment case does not contain any repeaters.). Therefore, $R_{\text{AD}} = \emptyset$, and any repeaters in the hierarchical case are replaced with sensors in the ad hoc case.

TABLE X
PARAMETERS USED IN VALIDATING MODELS

Parameter	Value	Comments
D	20	Rail trip duration in hours.
ζ	0.2	Probability of event occurrence during trip.
F_j	3×10^{-3}	Visibility requirement for probability of false alarm at a container.
E_j	0.85	Visibility requirement for probability of detection at a container.
TR_j	0.85	Visibility requirement for making a timely event report to decision makers.
α	1×10^{-3}	Probability of false alarm for each sensor.
ϵ	0.90	Probability of detection for each sensor.
φ	0.90	Probability of timely event reporting for each sensor.
l	690	Message length in bytes.
λ_i	9.0×10^{-2}	Message generation rate for a sensor. This results in 90 messages every 1,000 hours.
$\Pr(H)$	0.90	Probability of train being in cellular coverage.
$\Pr(I)$	0.90	Probability of train being in satellite coverage.
C_c	5×10^{-5}	Cost in units of sending one byte over a cellular link.
C_s	2×10^{-4}	Cost in units of sending one byte over a satellite link.
$C_{HL} + C_H$	46	Cost to acquire and install each sensor in the hierarchical case.
$C'_{HL} + C'_H$	51	Cost to acquire and install each sensor in the ad hoc case.
$C_A + C_{AL}$	101	Cost to acquire and install each repeater.
C_α	20000	Cost per false alarm.
	14.6	Amortized cost of backhaul communications device.

Using Fig. 5 as an example we assume, without loss of generality, that sensor 1 is assigned to container 1, sensor 2 is assigned to container 2, etc. Then, in the hierarchical deployment $R_{\text{Hier}} = \{6\}$, i.e., the repeater with id code 6 is assigned, while $R_{\text{AD}} = \emptyset$ in the ad hoc case. In addition, assume that in both the hierarchical and ad hoc cases $B_{\text{Hier}} = B_{\text{AD}} = \{9\}$.

- 2) Since we assume that the same visibility conditions hold in both cases, then we can conclude that the ad hoc deployment case contains at least as many sensors as the hierarchical case, with equality being achieved if the hierarchical case did not contain any repeaters. This condition is captured below:

$$|S_{\text{Hier}}| + |R_{\text{Hier}}| \leq |S_{\text{AD}}| \quad (26)$$

Referring to Fig. 5, the set of sensors assigned in the hierarchical case is $S_{\text{Hier}} = \{1, 2, 3, 4, 5, 7, 8\}$. The set of sensors assigned in the ad hoc case is $S_{\text{AD}} = \{1, 2, 3, 4, 5, 7, 8, 10\}$. Thus, we see

that the claim from equation (26) holds with equality.

- 3) The set of containers that that has sensors in the ad hoc deployment case, but which was not assigned sensors in the hierarchical case is defined as:

$$J_{AD} \setminus J_{Hier} \quad (27)$$

Observe that this set is empty if no additional containers are assigned sensors in the ad hoc deployment case. Similarly the set of communications devices used in the ad hoc deployment case, but not in the hierarchical case is defined as:

$$I_{AD} \setminus I_{Hier} \quad (28)$$

As with the containers, this set is empty if no additional communications devices are used in the ad hoc deployment case. Note that, since we assume that both cases contain just one backhaul communications device while the ad hoc deployment case contains no repeaters, then equation (28) simplifies to:

$$S_{AD} \setminus S_{Hier} \quad (29)$$

Using Fig. 5 as an example, then $J_{AD} \setminus J_{Hier} = \{6\}$ since container 6 is the only container that has a sensor assigned in the ad hoc deployment case, but which did not have a sensor in the hierarchical deployment. Similarly, $S_{AD} \setminus S_{Hier} = \{10\}$, since sensor 10 is the new sensor assigned in the ad hoc case.

- 4) From equations (14) and (21) we observe that false alarm and communications costs increase as additional sensors are added, while the savings lost due to missed detections and late event reports decrease. The cost metric of the ad hoc case is the cost metric of the hierarchical case plus the false alarm costs of any new sensors minus the costs of missed detection and untimely reporting due to the new sensors plus any savings from detecting and reporting an event in the desired notification window. To this sum we add the increase in communications costs for the new sensors as well as the installation and material costs for the new sensors. Finally, we subtract the material and installation costs of the repeaters and sensors that were included in the hierarchical deployment. This mapping is summarized in equation (25).

Returning to the example train shown in Fig. 5 let us assume that we are given the parameter values in Table X and that all the containers on the train have low values. Then, the cost metric for the initial hierarchical deployment is 14,178.4 units while the cost metric for the ad hoc deployment is 6,983.5 units. The following costs can be computed for the additional sensor in the ad hoc

deployment: false alarm cost for the additional sensor is 20 units, additional savings in event detection due to the new sensors is 3,600 units, savings resulting from decision maker notification in a timely manner is 3,600 units, the additional communication cost is approximately 0.11 units, cost of acquiring and installing the eight new sensors is 408 units, and the amount gained by not deploying a reader is 101 units. It can be shown that $6983.5 = 14178.4 + 20 - (3600 + 3600) + 0.11 + 408 - 322 - 101$, which confirms equation (25). ■

E. Container Placement

For the purposes of this research we assume that containers have been placed in fixed locations on the train such that the aerodynamic efficiency of the train is maximized. We assume that container placement is done using Lai *et al.*'s [4] method. Please consult [4] for details on the objective function and constraints for this container placement methodology.

V. MODEL GROWTH AND VALIDATION

In this section we review model validation and the growth of the sensor placement problem with train size. Model validation seeks to determine if a given mathematical abstraction matches a real system. This task is generally hard to accomplish. Kleindorfer *et al.* [9] provides a more complete discussion on validation of models, especially simulation models. By validating our models we can have greater confidence in the optimization results reported by our models.

A. Model Growth and Computational Complexity

In this subsection we examine the growth of our models with different problem inputs. The optimization models described in Section IV have been solved using the Bonmin [10] solver running on the NEOS optimization server [11], [12]. Both models have been run for trains with 7, 14, 20, 27, and 33 containers (this translates to 3, 6, 9, 12, and 15 units respectively). The computational complexity of our models depends on the number of variables and constraints, with the problem becoming more complex with more variables and constraints. The growth in the number of variables and constraints is summarized in Fig. 6. From Fig. 6a it is clear that the train-mounted and trackside models have about the same number of variables. Note that the trackside model with fixed train speeds has additional variables, e.g., sensor transmission range and sensor read time, that are not found in the train mounted model. From Fig. 6b we see that the number of constraints in all three models increases gradually with train size. This growth

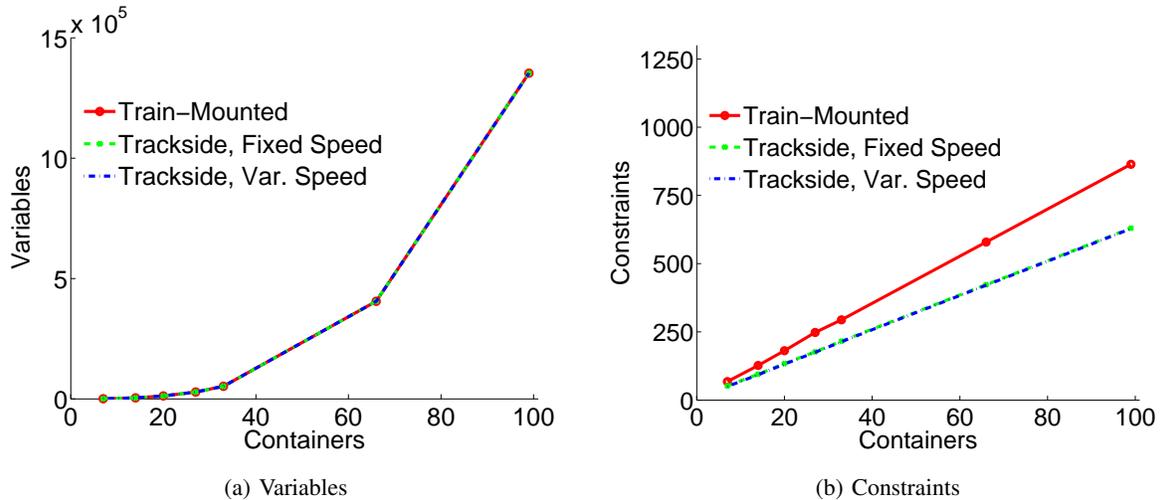


Fig. 6. Problem Growth in Number of Variables and Constraints

is partially due to the fact that there is one instance of equation (15) for every sensor and one instance each of equations (16), (19), and (20) for each container. The rapid growth in the number of variables motivates us to consider using heuristics to assign sensors and related communications infrastructure. In our future work we specify a heuristic for assigning sensors to containers in fixed positions on a train. In the rest of this paper and our future work we only consider the train-mounted and the trackside model with fixed train speeds.

B. Model Validation

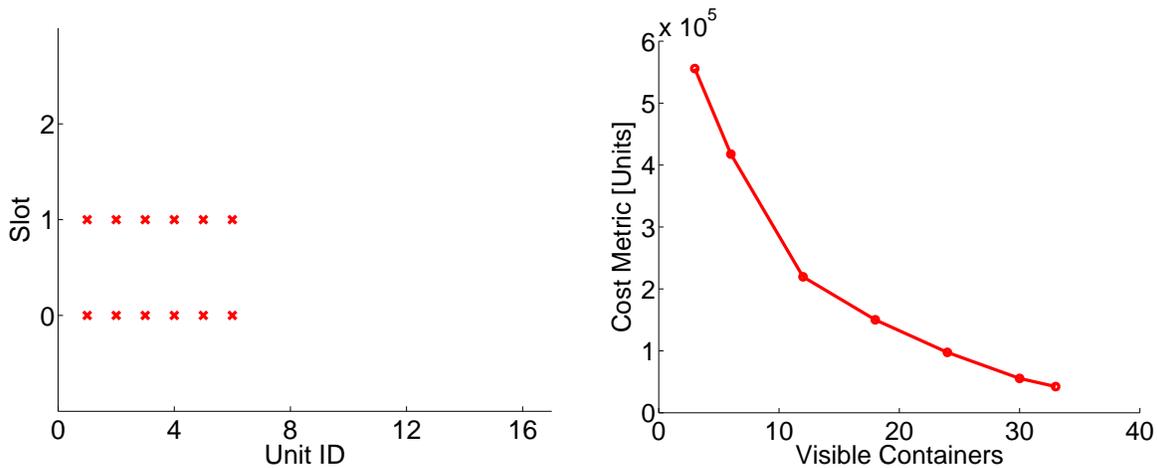
In this subsection we construct arguments for validating the train-mounted and trackside models by studying trends in the behavior of the optimization models at the boundaries of the visibility space. For the sake of discussion we will use an example train to illustrate our claims. We use the parameter values from Tables X and XI in our discussion.

1) *Train-Mounted Model*: Suppose we have a train with 15 units and 33 containers; where 20 of the containers have a low value, 9 have a medium value, and 4 have a high value. If the train-mounted model achieves an optimal result, it returns the cost metric at the optimal solution as well as the final sensor assignment.

Assume that there are initially enough sensors for each of the containers. Suppose that the visibility conditions on the containers are relaxed such that: $TR_j = 0.0$, $E_j = 0.0$, and $F_j = 1.0$, for some of the containers. In addition assume that there are exactly enough sensors available to satisfy the visibility constraints. Fig. 7a shows the slot and unit locations when only 12 of the 33 containers are visible.

TABLE XI
ADDITIONAL PARAMETERS USED IN VALIDATING MODELS

Parameter	Value	Comments
σ_j	200,000	Average savings resulting from event detection at high value container. Reference [13] indicates that in 2006 the average container entering the US had a value of 66,000.
σ_j	100,000	Average savings resulting from event detection at medium value container.
σ_j	20,000	Average savings resulting from event detection at low value container.



(a) Sensor Locations, where Locomotive = Unit 0, slot 0 is in the bottom level of the railcar, and slot 1 is in the top level.

(b) Visibility vs. Cost

Fig. 7. Train-mounted Model: Sensor Locations and Cost Metric Variation with Number of Visible Containers

Fig. 7b shows the relationship between the number of visible containers and the cost metric. As we have fewer sensors the cost metric per trip increases as more containers are not “protected” by any sensors.

As the rail trip duration is increased the cost metric per trip should increase as there is greater opportunity for messages to be transmitted. Fig. 8a shows that as the rail trip duration is increased the system cost metric also increases. Fig. 8b shows the relationship between the probability of event occurrence and the system cost metric. As events become more likely, the system cost metric per trip also increases. Figs. 7b and 8 show that the train-mounted model exhibits correct trends.

2) *Trackside Model with Fixed Speeds*: As stated in Sections III-B and IV-B the outputs of the trackside model include the system cost metric, sensor locations, maximum sensor read time, and minimum sensor

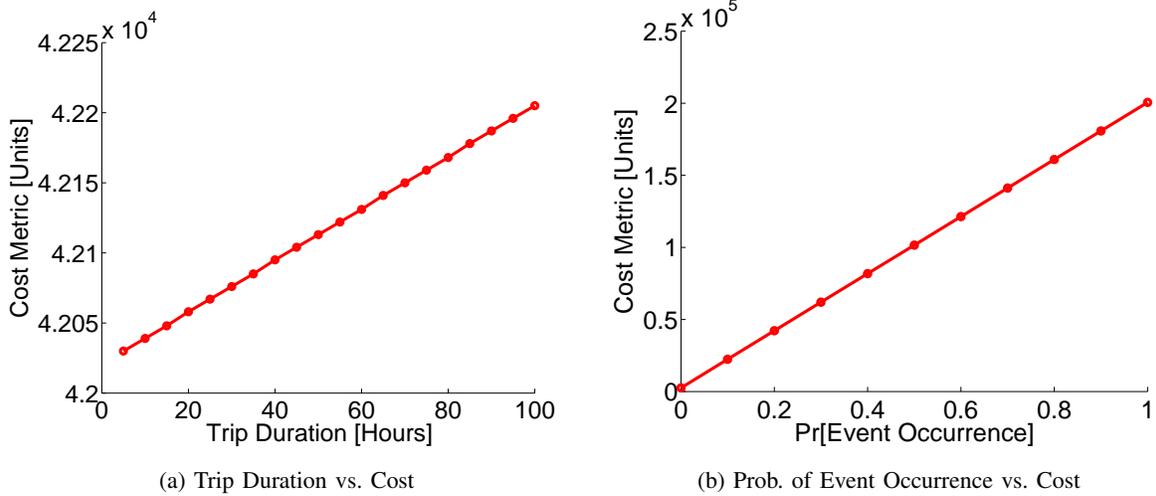


Fig. 8. Train-mounted Model: Trip Duration and Pr[Event Occurrence] versus Cost Metric

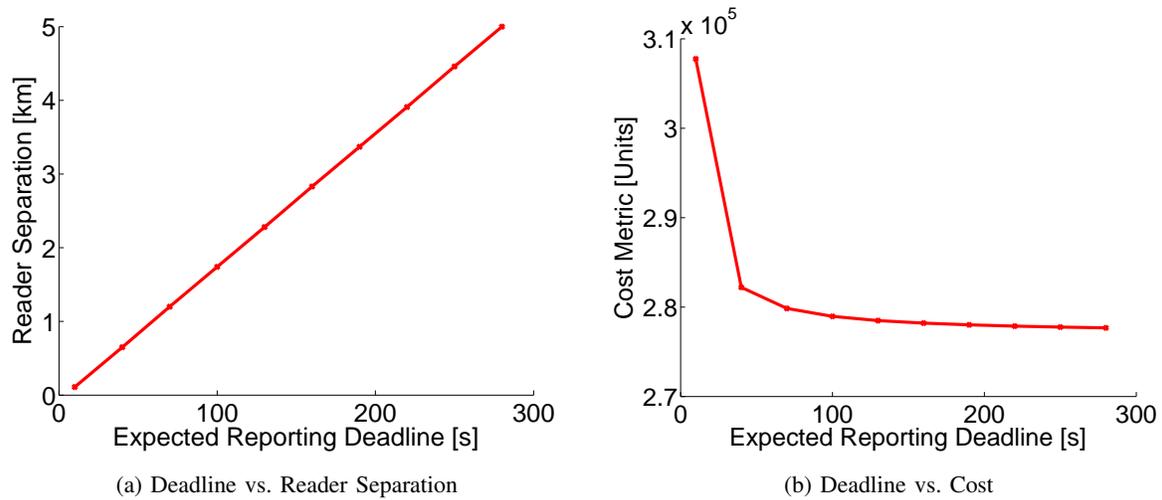


Fig. 9. Trackside Model: Reporting Deadline versus Reader Separation and Cost Metric

transmission range. In addition we also compute reader separation given the reporting deadline and probability of successful communications from a sensor to a trackside reader.

For the trackside model the cost metric for the entire system will increase, as was the case for the train-mounted model, as fewer sensors are available to be used on the train. This is because more and more of the containers are not protected by sensors. Assume that we have the same train configuration mentioned in Section V-B1, with each container being assigned a sensor while the readers are at the trackside.

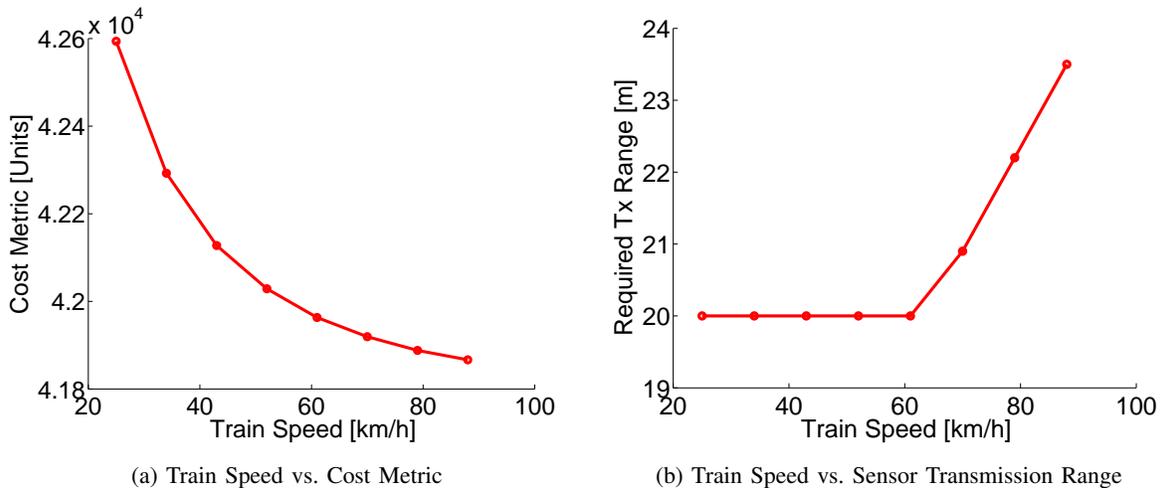


Fig. 10. Trackside Model: Train Speed versus Cost Metric and Sensor Transmission Range

Fig. 9 shows the effect of changes in the expected reporting deadline on reader separation and system cost metric when the train speed is fixed at 45 km/h. Fig. 9a shows the relationship between reporting deadline and reader separation. As the reporting deadline is reduced the trackside readers need to be placed closer together. Since more readers are required, the cost metric increases significantly as the reporting deadline is shortened. Fig. 9b shows the change in cost metric with the reporting deadline.

Fig. 10a shows that the cost metric decreases as the train speed is increased. As the train speed is increased the train can cover the distance between its origin and destination in a shorter time implying that the trackside readers can be placed further apart while satisfying the reporting deadlines. Finally, suppose that the system specifications state that each sensor is read in at most 3 s. As the train speed is increased equation (22) shows that the sensor transmission range must increase so that each sensor can be read in the specified interval. Fig. 10b shows that the sensor transmission range increases as expected. This relatively simple example shows that equation (22) correctly captures system operation for the trackside model.

In this section we have shown that our optimization models exhibit correct trends matching a real system. Therefore, we can have confidence in our results.

VI. RELATED WORK

In this section we provide an overview of solution techniques for mixed integer linear and mixed integer nonlinearly constrained problems; two classes of optimization problems that we have encountered in our modeling work.

A. Mixed Integer Linear Programs

A mixed integer linear program (MILP) is an optimization problem where the objective function and all of the constraints are linear functions, while some of the variables are integer-constrained [14]. Darby-Dowman and Wilson [15] state that integer program models are generally harder to solve than linear program models of the same size, while Bonami *et al.* [16] state that MILP are \mathcal{NP} -Hard problems. Mixed integer linear programs are either solved by branch-and-bound, branch-and-cut, or branch-and-price methods.

When solving integer programs a tree of the entire solution space is created, the root node of the tree is the entire state space, S , while all other nodes represent smaller partitions of the solution space. With branch-and-bound the branching is done by selecting a variable x with a fractional value k and then creating two sub-problems with the additional constraints $x \leq k$ and the other $x \geq k + 1$. This is called the Linear Programming Relaxation (LPR) At a selected node of the tree the integer program LPR is solved. If there is no feasible solution to the problem at that node, the node is eliminated. Otherwise if the solution of the linear programming relaxation is integer feasible and the objective function solution is less than the previous upper bound then the objective function value for this subproblem is set as the new upper bound for the objective function. Branching continues until the best integer feasible solution found is shown to be optimal. With branch-and-cut at each stage in the development of the solution space tree an equation called the cut is added to the set of constraints when carrying out the linear programming relaxation. The cut has the added requirement that it must not exclude any integer solutions at that node or any of its descendants; however, it may exclude integer solutions for preceding nodes. With branch-and-price an auxiliary problem is solved to identify which columns should be added to the linear programming relaxation. The relaxation is optimized and more columns are identified for addition to the LPR [15].

B. Mixed Integer Nonlinear Programs

A mixed integer nonlinear program (MINLP) is an optimization problem with some integer-constrained and continuous variables as well as nonlinear constraints and/or objective function. If all the variables are continuous, then we have a nonlinear program. MINLPs are a superset of mixed integer linear programs, where the reduction to MILP takes place when all of the functions in the optimization problem are linear [14].

Mixed Integer nonlinear programs are worse than \mathcal{NP} -Hard [14]. However convex MINLPs can be solved using the following techniques: branch-and-bound, extended cutting plane, outer approximation,

generalized Benders decomposition, LP/NLP-based branch-and-bound, and branch-and-cut [14], [16]. This section provides an overview of each of these techniques. More detailed explanations of the solution methods are found in [16]. Branch-and-bound for MINLPs is done just as for mixed integer linear programs, except that a nonlinear program is now solved at each node of the tree [14]. The extended cutting plane method constructs a mixed integer linear program relaxation and solves it. If the solution is not feasible, then a cutting plane of the most violated constraint at the optimal solution is added to the relaxation and the problem is re-solved and the process is repeated [14], [16]. Outer approximation (OA) is based on the observation that a MINLP is equivalent to a MILP of finite size. The MILP can be generated by linearizing both the objective and constraint functions. The linearized function is then solved and the integer solution from this step is used as a bound on the optimal value of the NLP. This process is repeated until the upper and lower bounds of the optimal value of the non-linear program are within a specified tolerance [16]. Generalized Benders decomposition is very similar to the outer approximation method except that it has only one continuous variable [14]. LP/NLP-based branch-and-bound is an extension of the OA method. It uses LPR to find an integer solution in a branch-and-bound tree and then solves the nonlinear program to get upper bounds on the solution [14], [16]. Branch-and-cut has been adapted to solving MINLPs [14]. This method is similar to branch-and-bound, but it adds cutting planes at each node of the tree to strengthen the NLP relaxation [14].

VII. CONCLUSION

This paper presented two models that can be used to find the optimal cost metric for a rail-borne cargo monitoring system. We presented the parameters and variables for our models. The models presented in Section IV are suitable to enable quantitative evaluation of the trade-offs that can be made when monitoring rail-borne cargo. In addition this paper has also shown that we a hierarchical deployment of sensors can be mapped to an ad hoc sensor assignment, given that the sensors assigned in the initial case are not moved to other containers. Finally, this paper has shown that there is a large number of variables involved in the models for sensor assignment. As a result, future work will determine if heuristics can yield near-optimal performance for sensor assignment.

ACKNOWLEDGMENTS

This work was supported in part by Oak Ridge National Laboratory (ORNL)—Award Number 4000043403. This material is also partially based upon work supported while V. S. Frost was serving at the National Science Foundation.

REFERENCES

- [1] European Conference of Ministers of Transport, *Container Transport Security Across Modes*. Paris, France: Organisation for Economic Co-operation and Development, 2005.
- [2] KC SmartPort. (2008, Nov. 10) Trade Data Exchange—Nothing short of a logistics revolution. Digital magazine. [Online]. Available: <http://www.joc-digital.com/joc/20081110/?pg=29>
- [3] Federal Bureau of Investigation. (2006, Jul. 21) Cargo Theft's High Cost. Headline. Federal Bureau of Investigation. [Online]. Available: http://www.fbi.gov/page2/july06/cargo_theft072106.htm
- [4] Y.-C. Lai *et al.*, "Optimizing the Aerodynamic Efficiency of Intermodal Freight Trains," *Transportation Research Part E: Logistics and Transportation Review*, vol. 44, no. 5, pp. 820–834, Sep. 2008.
- [5] Intermodal Committee, *Loading Capabilities Guide*, Association of American Railroads Std., Jun. 26 2003. [Online]. Available: <http://www.aar.org/AARPublications/~media/AARPublications/FreePubs/AAR%20Loading%20Capabilities%20Guide.ashx>
- [6] D. T. Fokum *et al.*, "Experiences from a Transportation Security Sensor Network Field Trial," University of Kansas, Lawrence, KS, Tech. Rep. ITTC-FY2009-TR-41420-11, Jun. 2009.
- [7] M. Kuehnhausen and V. S. Frost, "Application of the Java Message Service in Mobile Monitoring Environments," University of Kansas, Lawrence, KS, USA, Tech. Rep. ITTC-FY2010-TR-41420-18, Dec. 2009.
- [8] S. Nadarajah, "A Review of Results on Sums of Random Variables," *Acta Applicandae Mathematicae: An International Survey Journal on Applying Mathematics and Mathematical Applications*, vol. 103, no. 2, pp. 131–140, Sep. 2008.
- [9] G. B. Kleindorfer *et al.*, "Validation in Simulation: Various Positions in the Philosophy of Science," *Manage. Sci.*, vol. 44, no. 8, pp. 1087–1099, Aug. 1998.
- [10] P. Bonami. (2010, May) Bonmin. Project wiki. [Online]. Available: <https://projects.coin-or.org/Bonmin>
- [11] (2010, Feb.) NEOS Solvers. Solver listing. Argonne National Labs. Argonne, IL, USA. [Online]. Available: <http://neos.mcs.anl.gov/neos/solvers/index.html>
- [12] J. Czyzyk *et al.*, "The NEOS Server," *Comput. Sci. Eng., IEEE*, vol. 5, no. 3, pp. 68–75, Jul.–Sep. 1998.
- [13] S. E. Flynn. (2006, Jan.–Feb.) Port Security Is Still a House of Cards. Article. Council on Foreign Relations. [Online]. Available: <http://www.cfr.org/publication/9629/>
- [14] A. N. Letchford, "Mixed-Integer Non-Linear Programming: A Survey," presented at the 1st LANCS Workshop on Discrete and Non-Linear Optimisation, Southampton, United Kingdom, Feb. 2009.
- [15] K. Darby-Dowman and J. M. Wilson, "Developments in Linear and Integer Programming," *The Journal of the Operational Research Society, Special Issue: Applications and Developments in Mathematical Programming*, vol. 53, no. 10, pp. 1065–1071, Oct. 2002. [Online]. Available: <http://www.jstor.org/stable/822966>
- [16] P. Bonami *et al.*, "Algorithms and Software for Convex Mixed Integer Nonlinear Programs," Computer Sciences Department, University of Wisconsin-Madison, Madison, WI, USA, Tech. Rep. 1664, Oct. 2009.