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

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 Ahlgren *et al.*'s concept of invariants, which are components of a system that cannot be changed without losing backward compatibility. Our review show that while several sensor network architectures exist, each with different attributes, these architectures share several invariants.

CONTENTS

I Introduction 1

II Related Work and Context for Discussion 1

II-A Issues with Existing Architectures . . . 4

III Architecture Taxonomy 5

III-A Architecture Classification . . . 5

III-B Standards-Based Sensor Networks . . . 10

III-C Internet-Connected Sensor Networks . . . 12

III-D Context-Aware Sensor Networks . . . 14

III-E Agent-Based Sensor Networks . . . 15

III-F Service-Oriented Sensor Networks . . . 16

III-G Secure and Fault-Tolerant Sensor Networks . . . 18

III-H Vehicle-Based Sensor Networks . . . 19

III-I Habitat-Monitoring Sensor Networks . . . 21

III-J Multi-Owner Sensor Network Architecture (MOSN) . . . 21

IV Architecture Comparison Summary 22

V Conclusion 22

References 24

LIST OF TABLES

I Summary of Architecture Features . . . 23

II Summary of Architecture Invariants . . . 28

I. 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 to a base station, or other central site. In general, a wireless sensor node (WSN) consists 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 [1]. Base stations, like 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 [1].

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 has been studied in [2], [3], and [4], while deployment experiences are recorded in [4]–[9]. 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 [10], which are system elements that cannot be changed without losing backward compatibility.

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

II. 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 [10]. 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 of these characteristics should help us determine the quality of the given architecture [10].

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 assumed 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 [5], [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 [4], [8], [9], [18], [21], [22]) or different layers of programs (as in [23]).

In this paper we classify sensor networks in terms of the attributes above, and determine 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 should 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 noncooperative. With cooperative communications, sensors cooperate with other sensors to fulfill the observer's need; noncooperative 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.

A. 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 to only two types of sensor network architectures, we contend that the variety 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 III 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.

III. 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 II. For each architecture we will also classify each of its attributes in terms of invariants, as introduced in Section II.

A. Architecture Classification

In Section II 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 [4], [8], and [9]. 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 [4]. The reader may consult Fig. 2 in [4], Fig. 1 in [8], or Fig. 1 in [9] for a system architecture diagram.

Reference [9] goes beyond the simple architecture presented in [4] to present an architecture that organizes all the sensor nodes within a sensor patch into a routing tree. In addition, computation is located within the sensor network so as to reduce the energy consumption of the individual nodes as well as to reduce the volume of data being transmitted. Here, the sensor network also has an independent verification network the sole purpose of which 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, [9] also gives examples of sensor networks whose architectures are extensions of the basic architecture presented in [4], [8], and [9]. 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. The sensor nodes have an SQL-variant query interpreter running on each node, and sensor nodes receive queries in an epidemic fashion [9]. The key attribute for this family of architectures is the tiered architecture. An additional attribute for the Tiny Diffusion architecture is support for data diffusion; 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 enroute. 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 assumes 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 key nodes. Observations refer to readings from sensors, while certain collections of observations constitute an event, e.g., an 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: multiresolution 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 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 multitier 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 Fig. 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, attribute-based naming, and location awareness. Hierarchical clustering allows sensor nodes to aggregate into clusters, while attribute-based naming allows users to query the sensor network by some attribute, e.g., what is the average temperature in a given quadrant. Finally, location awareness 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 messages, 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 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 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—fusion channels—used for fusion functions. 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, 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 and application layers 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. A query optimizer is located on the sensor network's gateway node. The query optimizer generates a distributed query processing plan for queries generated from outside 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 have to be submitted to the query optimizer node in a network.

B. 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 II-A 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] and [36].

Reference [37] makes the case for the use of standards in sensor networks, particularly 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. Lee and Reichardt [37] go 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 IEEE 1451 [38] and Universal Plug and Play (UPnP) [39], 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 geoprocessing framework. 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, 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 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 across a network using a 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. 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 Fig. 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, and 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 II we can also conclude that IP is an invariant for this architecture.

C. 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 multihop 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 Fig. 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). IrisNet aims to provide software components for a world-wide sensor network [5], [6]. Gibbons *et al.* [5], [6] 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 queriable 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 services 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 [5]. Please consult Fig. 1 in [5] 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 II 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[1] 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.
- 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]. Fig. 2 in [17] shows the extended architecture for sensor networks

[1]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.

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 are comprised 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 formed an Internet Engineering Task Force (IETF) working group to study routing over low-power and lossy networks, such as sensor networks [42].

D. 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]. Fig. 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.

E. Agent-Based Sensor Networks

Agents can also be used in designing sensor networks, as we have already seen from [5]. 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 intercluster 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 retasking 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]. Fig. 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.

F. 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 (SOSANETs) are a type of customizable SANET.

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 deployed more rapidly than sensor networks based on TinyDB since queries are automatically discovered under TinySOA. Fig. 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 integrated with Radio Frequency Identification (RFID) and used to 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 back-end, 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.
- Notification manager, which uses a web service to distribute event messages.

The gateway (platform abstraction) layer facilitates interoperability between sensor platforms. 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 executables 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. Fig. 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.

G. 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 [46], but this paper presents a scheme for enhancing the reliability of sensor networks. When a sink has little energy left, it is relocated to another sensor node [46].

Reference [47] presents an architecture for a secure and survivable wireless sensor network with heterogeneous nodes. The architecture will provide security and survivability mechanisms, techniques, 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 [47] to obtain more details on the architecture. It should be noted that [47] 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. Fig. 1 from [47] 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.

H. Vehicle-Based Sensor Networks

In the near future, we will begin to see sensor networks deployed to vehicles to enhance driver safety, and to 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) such that 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 Hypertext Transfer Protocol (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), that is, 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]. Fig. 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 [7]. This paper discusses a mobile sensor network composed of CarTel nodes that processes heterogeneous data. In general, mobile sensor networks allow for coverage of a larger surface area with fewer sensors. Each CarTel node consists of a mobile embedded 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 application program interface (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, Network, and Mule Adaptation layers. The CafNet network layer supports buffering of some data [7]. Fig. 2 in [7] 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 intervehicle communication is different from traditional mobile ad hoc networks (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 (on-board device) 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 Fig. 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 intercar train communications and 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.

I. Habitat-Monitoring Sensor Networks

As we observed in Section III-A, some of the earliest sensor networks were used for monitoring seabirds on Grand Duck Island in Maine [4], [8], and [9]. Reference [8] 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 iPaqs (known as gizmos in the paper) to allow for remote management of the sensor network [8].

J. Multi-Owner Sensor Network Architecture (MOSN)

There is growing literature concerning the architecture and design of sensor networks [1]–[3], 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 [4].

A premise of this discussion is that elements of the sensor network will be owned by multiple organizations and will 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].

The MOSN architecture extends ORNL's SensorNet Information Architecture and has been built upon the existing sensor network architectures (e.g., [5], [51], [53], and [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 that 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 commercial off-the-shelf (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] and [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 observe that the key attributes of the multi-owner sensor network architecture include: Internet connectivity support, security support, and 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.

IV. 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 I summarizes the key features, while Table II summarizes the invariants of the architectures presented in Section III.

V. 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 architecture, as was previously argued in [11]

TABLE I

SUMMARY OF ARCHITECTURE FEATURES

Architecture	Features
Habitat monitoring system [4], [8], [9]	Internet connectivity support and tiered
Directed Diffusion [28]	Fusion support
Data-centric Storage [29]	Location encoding
TSAR [18]	Metadata communications and tiered
Middleware Design for Integration of Sensor Networks and Mobile Devices [22]	Tiered
DIMENSIONS [30]	Location encoding and tiered
DFuse [33]	Fusion support
Cougar [34]	Fusion support and tiered
ORNL SensorNet [35], [37]	Fault tolerant, Internet connectivity support, location encoding, security support, standards-based, and tiered
WINS [41]	Internet connectivity support and tiered
IrisNet [5], [6]	Agent-based, Internet connectivity support, and tiered
Janus [17]	Agent-based and Internet connectivity support
CASN [43]	Context-aware and tiered
MADSN and MAWSN [12]	Agent-based and tiered
SOSANET [44]	Service-oriented
Multiplatform Wireless Sensor Network [45]	Service-oriented and tiered
Secure and Survivable Wireless Sensor Networks [45]	Security support
VITP [21]	Location encoding and tiered
CarTel [7]	Delay tolerant and location encoding
TrafficView [49]	Location encoding
Multi-owner sensor network architecture [23]	Internet connectivity support, service oriented, security support, standards-based, and tiered

and [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 [10]) 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 functionalities needed by sensor network designers. Such a design might allow better portability of code and ideas from one sensor network to the next.

ACKNOWLEDGMENTS

This work was supported in part by Oak Ridge National Laboratory (ORNL)—Award Number 4000043403.

The authors would like to thank A. Francis for reading and commenting on a previous version of this paper.

REFERENCES

[1] F. Zhao and L. Guibas, *Wireless Sensor Networks: An Information Processing Approach.* San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2004.

[2] D. Estrin *et al.*, "Connecting the Physical World with Pervasive Networks," *IEEE Pervasive Computing*, vol. 1, no. 1, pp. 59–69, Jan-Mar 2002.

[3] I. F. Akyildiz *et al.*, "Wireless Sensor Networks: a Survey," *Computer Networks*, vol. 38, no. 4, pp. 393–422, 2002.

[4] R. Szewczyk *et al.*, "Lessons From a Sensor Network Expedition," in *EWSN 2004: Proceedings of the First European Workshop on Sensor Networks*, Jan. 2004.

[5] P. B. Gibbons *et al.*, "IrisNet: an Architecture for a Worldwide Sensor Web," *IEEE Pervasive Computing*, vol. 2, no. 4, pp. 22–33, Oct.-Dec. 2003.

[6] J. Campbell *et al.*, "IrisNet: an Internet-scale Architecture for Multimedia Sensors," in *MULTIMEDIA '05: Proceedings of the 13th Annual ACM International Conference on Multimedia*. Singapore: ACM, 2005, pp. 81–88.

[7] B. Hull *et al.*, "CarTel: a Distributed Mobile Sensor Computing System," in *SenSys '06: Proceedings of the 4th International Conference on Embedded Networked Sensor Systems*. Boulder, CO, USA: ACM Press, 2006, pp. 125–138.

[8] A. Mainwaring *et al.*, "Wireless Sensor Networks for Habitat Monitoring," in *WSNA '02: Proceedings of the 1st ACM international workshop on Wireless Sensor Networks and Applications*. Atlanta, GA, USA: ACM Press, 2002, pp. 88–97.

[9] R. Szewczyk *et al.*, "Habitat Monitoring with Sensor Networks," *Commun. ACM*, vol. 47, no. 6, pp. 34–40, June 2004.

[10] B. Ahlgren *et al.*, “Invariants: a New Design Methodology for Network Architectures,” in *FDNA ’04: Proceedings of the ACM SIGCOMM Workshop on Future Directions in Network Architecture*. Portland, OR, USA: ACM Press, 2004, pp. 65–70.

[11] D. Culler *et al.*, “Towards a Sensor Network Architecture: Lowering the Waistline,” in *HOTOS’05: Proc. 10th Conference on Hot Topics in Operating Systems*, USENIX Association. Santa Fe, NM, USA: USENIX Association, June 2005, pp. 24–30.

[12] M. Chen *et al.*, “Applications and Design Issues for Mobile Agents in Wireless Sensor Networks,” *IEEE Wireless Communications [see also IEEE Personal Communications]*, vol. 14, no. 6, pp. 20–26, Dec 2007.

[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*. Karlsruhe, Germany: ACM, 2003, pp. 27–34.

[14] D. Hutchison and J. P. G. Sterbenz. (2007, Feb. 6) ResiliNets Architecture Definitions. Wiki. The Information and Telecommunication Technology Center. [Online]. Available: http://wiki.ittc.ku.edu/resilinets_wiki/index.php/Definitions

[15] R. J. Abbott, “Resourceful Systems for Fault Tolerance, Reliability, and Safety,” *ACM Comput. Surv.*, vol. 22, no. 1, pp. 35–68, 1990.

[16] W. Chen and 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 *et al.*, “Janus: an Architecture for Flexible Access to Sensor Networks,” in *DIN ’05: Proceedings of the 1st ACM Workshop on Dynamic Interconnection of Networks*. Cologne, Germany: ACM Press, 2005, pp. 48–52.

[18] P. Desnoyers *et al.*, “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*. San Diego, CA, USA: ACM Press, Sep 2005, pp. 39–50.

[19] A. Perrig *et al.*, “SPINS: Security Protocols for Sensor Networks,” in *MobiCom ’01: Proc. of the 7th Annual Int’l Conference on Mobile Computing and Networking*. New York, NY, USA: ACM, 2001, pp. 189–199.

[20] G. Biegel and V. Cahill, “A Framework for Developing Mobile, Context-aware Applications,” in *PerCom 2004: Proc. of the Second IEEE Annual Conf. Pervasive Computing and Communications*. Orlando, FL, USA: IEEE Computer Society, 14-17 Mar 2004, pp. 361–365.

[21] M. D. Dikaiakos *et al.*, “VITP: an Information Transfer Protocol for Vehicular Computing,” in *VANET ’05: Proceedings of the 2nd ACM International Workshop on Vehicular Ad Hoc Networks*. Cologne, Germany: ACM Press, 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*. Grenoble, France: ACM Press, 2005, pp. 1–5.

[23] P. Mani *et al.*, “A Unified Architecture for Sensor Networks with Multiple Owners,” in *ACM SenSys 2008*, Submitted.

[24] J. Feng *et al.*, “System-Architectures for Sensor Networks Issues, Alternatives, and Directions,” in *ICCD’02: IEEE International Conference on Computer Design*, IEEE. Freiburg, Germany: IEEE Computer Society, Sep 2002, pp. 226–231.

[25] S. Tilak *et al.*, “A Taxonomy of Wireless Micro-sensor Network Models,” *SIGMOBILE Mobile Computing and Communications Review*, vol. 6, no. 2, pp. 28–36, Feb 2002.

[26] F. Martincic and L. Schwiebert, *Handbook of Sensor Networks: Algorithms and Architectures*. Hoboken, NJ: John Wiley & Sons, 2005, ch. Introduction to Wireless Sensor Networking, pp. 25–26.

[27] V. Handziski *et al.*, “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). Berlin, Germany: Technische Universitat Berlin, July 2003, Tech. Report, pp. 10–17.

[28] J. Heidemann *et al.*, "Building Efficient Wireless Sensor Networks with Low-level Naming," in *SOSP '01: Proceedings of the 18th ACM Symposium on Operating Systems Principles*. Banff, AB, Canada: ACM, 2001, pp. 146–159.

[29] S. Shenker *et al.*, "Data-centric Storage in Sensornets," *SIGCOMM Computer Communications Review*, vol. 33, no. 1, pp. 137–142, Jan 2003.

[30] D. Ganesan *et al.*, "DIMENSIONS: Why Do we Need a New Data Handling Architecture for Sensor Networks?" *SIGCOMM Computer Communication Review*, vol. 33, no. 1, pp. 143–148, Jan 2003.

[31] K. Römer *et al.*, "Middleware Challenges for Wireless Sensor Networks," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 6, no. 4, pp. 59–61, 2002.

[32] C.-C. Shen *et al.*, "Sensor Information Networking Architecture and Applications," *IEEE Personal Communications, [see also IEEE Wireless Communications]*, vol. 8, no. 4, pp. 52–59, Aug 2001.

[33] U. Ramachandran *et al.*, "Dynamic Data Fusion for Future Sensor Networks," *ACM Transactions on Sensor Networks (TOSN)*, vol. 2, no. 3, pp. 404–443, Aug 2006.

[34] Y. Yao and J. Gehrke, "The Cougar Approach to in-network Query Processing in Sensor Networks," *SIGMOD Rec.*, vol. 31, no. 3, pp. 9–18, Sep 2002.

[35] B. L. Gorman *et al.*, "Advancing Sensor Web Interoperability," *Sensors Magazine*, vol. 22, no. 4, pp. 14–18, 2005. [Online]. Available: http://www.sensorsmag.com/sensors/article/articleDetail.jsp?id=185897

[36] G. Percivall and C. Reed, "OGC Sensor Web Enablement Standards," *Sensors and Transducers*, vol. 9, no. 9, pp. 698–706, Sep 2006.

[37] K. B. Lee and M. E. Reichardt, "Open Standards for Homeland Security Sensor Networks," *IEEE Instrumentation & Measurement Magazine*, vol. 8, no. 5, pp. 14–21, Dec 2005.

[38] *A Smart Transducer Interface for Sensors and Actuators*, IEEE Draft Std., IEEE Std. 1451, 2007.

[39] *UPnP Device Architecture*, UPnP Forum Std. v. 1.0, 2006.

[40] Computational Sciences and Engineering Division, "SensorNet: Concept Definition Document," Oak Ridge National Laboratory, Tech. Report, 2004.

[41] G. J. Pottie and W. J. Kaiser, "Wireless Integrated Network Sensors," *Communications of the ACM*, vol. 43, no. 5, pp. 51–58, May 2000.

[42] J. P. Vasseur. (2007, Dec. 17) Routing Over Low Power and Lossy Networks (roll). IETF Working Group. IETF. [Online]. Available: http://www.ietf.org/html.charters/roll-charter.html

[43] Q. Huaifeng and Z. Xingshe, "Context Aware Sensornet," in *MPAC '05: Proceedings of the 3rd International Workshop on Middleware for Pervasive and Ad-hoc Computing*. Grenoble, France: ACM Press, 2005, pp. 1–7.

[44] A. Rezgui and M. Eltoweissy, "Service-Oriented Sensor-Actuator Networks," *IEEE Communications Magazine*, vol. 45, no. 12, pp. 92–100, Dec 2007.

[45] M. Marin-Perianu *et al.*, "Decentralized Enterprise Systems: a Multiplatform Wireless Sensor Network Approach," *IEEE Wireless Communications [see also IEEE Personal Communications]*, vol. 14, no. 6, pp. 57–66, Dec 2007.

[46] I. Saleh *et al.*, "In-network Fault Tolerance in Networked Sensor Systems," in *DIWANS '06: Proc. of the 2006 Workshop on Dependability Issues in Wireless Ad Hoc Networks and Sensor Networks*. New York, NY, USA: ACM, 2006, pp. 47–54.

[47] Y. Qian *et al.*, “A Design for Secure and Survivable Wireless Sensor Networks,” *IEEE Wireless Communications [see also IEEE Personal Communications]*, vol. 14, no. 5, pp. 30–37, Oct 2007.

[48] I. Chisalita and N. Shahmehri, “A Peer-to-peer Approach to Vehicular Communication for the Support of Traffic Safety Applications,” in *Proc. IEEE 5th Int’l Conf. Intelligent Transportation Systems*, Sep 2002, pp. 336–341.

[49] T. Nadeem *et al.*, “TrafficView: Traffic Data Dissemination Using Car-to-Car Communication,” *SIGMOBILE Mobile Computing and Communications Review*, vol. 8, no. 3, pp. 6–19, July 2004.

[50] 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.

[51] S. Muralidharan *et al.*, “SensorNet Architecture with Multiple Owners,” University of Kansas, Lawrence, Kansas, Tech. Report ITTC-FY2008-TR-41420-02, July 2007.

[52] G. J. Minden *et al.*, “Architecture and Prototype of an Ambient Computational Environment: Final Report,” University of Kansas, Lawrence, Kansas, Tech. Report ITTC-FY2004-TR-23150-09, July 2003.

[53] M. Botts *et al.* (2006) OGC Sensor Web Enablement: Overview and High Level Architecture, OGC 06-050r2. Architecture. Open Geospatial Consortium. [Online]. Available: http://www.opengeospatial.org/pt/06-046r2

[54] M. Botts, *OpenGIS Sensor Model Language (SensorML) Implementation Specification*, Specification, Open Geospatial Consortium Draft proposed version OGC 05-086r2, Rev. 1.0, 2005. [Online]. Available: 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.

TABLE II

SUMMARY OF ARCHITECTURE INVARIANTS

Architecture Classification	Invariants	
	Explicit	Implicit
Habitat monitoring system [4], [8], [9]	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	
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 [5], [6]	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
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 [7]	CafNet	
TrafficView [49]		Location encoding scheme
Multi-owner sensor network architecture [23]		Service-oriented architecture, standards-based architecture, and IP