

## SCIMPS: AN INTEGRATED APPROACH TO DISTRIBUTED PROCESSING IN SENSOR WEBS

David Andrews, Joe Evans, Venumadhav Mangipudi, Aditya Mandapaka

*Information Technology and Telecommunication Center, Department of Electrical Engineering and Computer Science, University of Kansas  
{dandrews, evans, madhav, aditya}@itc.ukans.edu*

**Abstract:** This paper presents a new tightly coupled computation/communication design developed to support the unique operational requirements of sensor webs. A critical challenge of sensor webs is supporting the ad hoc deployment of unknown numbers of nodes. KUASAR is a new routing algorithm developed for the unique operational modes of sensor webs. The basic operation of KUASAR is presented along with its performance capabilities in the presence of nodes being randomly added and deleted from the sensor web. The key performance capabilities and correctness of KUASAR are also presented.

**Keywords:** Sensor Webs, Routing Protocols, Computer Architecture, Networking

### 1. INTRODUCTION

Conceptually sensor web systems can be viewed as functional info-spheres of computation, where hundreds of thousands of sensors are data producers, distributed autonomous agent processing algorithms are knowledge generators, timeliness responses are specified on dynamic, non-stochastic knowledge states, and distributed actuators co-ordinate to achieve an enhanced system response. (Hill *et. al.*, 2000)(Culler *et. al.*, 2001) characterizes this new genre of embedded software as being agile, self-organizing, critically resource constrained, and communication-centric on numerous small devices operating as a collective. The operational mode is intensely concurrent with low burst rates projected in the 1-10 hertz range. Additionally, power considerations will be a driving factor in the realization of these systems (Wang, Chandrakasan, 2001). Current goals include 5 mW/MIP power factors, orders of magnitude lower than current techniques and technology can provide.

The quality of a response is both a function of the timeliness, and the quality of knowledge at an instant in time. We can view this model as a course

decomposition of the abstract domains that comprise this model such as shown in fig. 1.

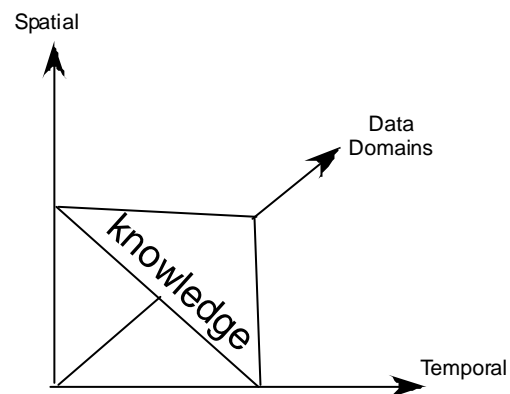


Fig. 1. Course Decomposition of Embedded Systems Functional Domains

Computational nodes perform demand driven pass-through conditioning of sensor data, and engage in dynamic knowledge formation via data sharing and processing across distributed platforms. This represents a data-centric model, with focus on

applying operations to data fields and sets. The formation of knowledge is based on retrieval and combination of data across the network in a spatial domain, and timeliness of data updates. Primitive operations, such as domain data selection, reduction and combination are required to support processing within this new paradigm. This is fundamentally opposite to non-embedded models that have "flat" spatial locality and no temporal or domain specific planes. Temporal requirements result from sensor input rates as well as timeliness constraints on the knowledge production and response. The computational model should elevate timeliness information up into the application domain where the relationship between time, precision, and quality of service are best understood.

Researchers have been investigating new ad hoc routing protocols that best support the unique operational requirements of sensor webs (Royer and Toh, 1999). Of particular consideration is the ability to form networks of nodes with no a-priori knowledge of numbers or position. The life cycle of a node is fundamentally dynamic, with new nodes being added into the network, and existing nodes becoming unavailable due to limited power. Most researchers are attempting to balance power consumption and node life by forming nearest neighbor interconnections, and periodically updating routing and connectivity matrices.

Clustering and domain selection routing algorithms have been proposed that allow individual sensors to engage and disengage from the network in a power aware fashion. The Sensor Information Networking Architecture (SINA) (Rentalá *et al.*, 1999) forms hierarchical clusters of autonomous groups of nodes. This clustering process can be applied recursively to form a hierarchy of clusters. Nodes are queried for information based on attributes, where complex queries can be formed with little overhead within the network. In contrast to SINA, a Self-organizing Medium Access Control for Sensor networks (SMACS) (Sohrabi *et al.*, 2000) has been proposed that enables nodes to discover neighbors without the need for master nodes. The SMACS architecture builds a flat topology with no clusters or cluster heads. The eavesdrop and register (EAR) (Sohrabi *et al.*, 2000) architecture has been developed for communication between mobile nodes and stationary nodes on the ground. To conserve energy, mobile nodes keep a registry of all sensors within a neighborhood and make handoff decisions when the SNR drops below a pre-determined threshold value. During a boot up period, invitation messages are broadcast as a trigger. Each mobile node eavesdrops and forms a registry of all stationary nodes within hearing range. Most proposed routing protocols fall into two main categories: flat routing protocols, and hierarchical protocols. The objective of all routing protocols is to limit node to node communications between pairs of near nodes to reduce power.

The sequential assignment flat routing protocol (SAR) (Rentalá *et al.*, 1999) builds multiple routes between a sink and source. This is to minimize the

time and cost of computing new routes during failures. A routing tree is built outwards from the sink nodes that attempt to minimize the use of low Quality of Service (QoS) and energy reserves. Each node belongs to multiple paths and each sensor can control which one-hop neighbor of the sink to use for messaging. The SAR algorithm uses an adaptive QoS metric and a measure of energy resources to arrive at an additive QoS metric and a weight coefficient associated with a packet priority level. During system operation, the SAR algorithm attempts to minimize the average weighted QoS metric. This algorithm requires re-computing paths to account for changes in network topology.

Directed diffusion is another flat routing protocol proposed by (Intanagonwiwat *et al.*, 2000) that allows sensor data to be named. Sink nodes query the sensor web based on data names, and sensor nodes may selectively respond. Intermediate nodes may route data from sensors towards the sink node. (Chen *et al.*, 2001) proposed a minimum cost forwarding algorithm for large sensor networks. In this approach, each node contains a least cost estimate between itself and the sink node. Each message in the system is broadcast, and intermediate nodes check to see if they are on the least cost path. If so, the node re-broadcasts the message. (Kulik *et al.*, 1999) proposed the Sensor Protocols for Information via Negotiation (SPIN). These protocols disseminate individual sensor information to all sensor nodes under the assumption that all are potential sinks. This algorithm uses negotiation and information descriptors to overcome the potential information implosion that can be caused by flooding the network with messages sent to all nodes.

(Heinzelman *et al.*, 2000) proposed a low energy adaptive clustering hierarchy (LEACH) routing protocol as an energy efficient communication protocol for wireless sensor networks. In LEACH, self-elected cluster heads collect data from all sensor nodes in a cluster, and use data fusion methods to aggregate and transmit the data directly to the base station. The appointment of a cluster head is made periodically with the self-appointed cluster head announcing its role to its neighbors. This scheme reduces power by limiting all communications from within a cluster to the head node. This algorithm requires additional energy expenditure for periodic exploration of connectivity between nodes, and the periodic re-assignment of the cluster head.

## **2. KU AD HOC SENSOR WEB ACTIVE ROUTING (KUASAR)**

KUASAR is a routing protocol that has been designed to support attribute-based naming and formation of knowledge domains. Additionally, KUASAR promotes the integration of communications and computations in sensor web nodes to achieve a lightweight on-demand routing protocol for ad hoc sensor webs. KUASAR allows the sink to initiate data flow in the network. Additionally KUASAR lets sensor nodes relay data

back to the sink reliably and in the shortest possible multi-hop path. KUASAR converges quickly without the use of any explicit routing and control messages. KUASAR also has the ability to adapt very quickly to changes in topology avoiding/detecting loops without the use of any control packets.

### 2.1 KUASAR - Theory and Motivation

Wireless sensor webs consist of nodes that have restricted capabilities of processing and communication. These sensor webs have a special many-to-one traffic pattern not common to other wireless networks – most nodes are sources of data that transmit to a sink, where the processing takes place. Quite clearly, multi-hop shortest-path routing lends itself as a good routing strategy for such networks of homogeneous nodes with limited range. KUASAR advocates the principle of multi-hop shortest-path routing, and keeps our objectives of tightly coupling the computation/communication.

Section 2.2 discusses *active routing*. Section 2.3 presents the basic phases and algorithms. Section 3 further discusses the properties of *active routing* and investigates key attributes and issues (shortest path, looping) pertaining to the algorithm.

### 2.2 Active Routing

A simple sensor network can be represented as a set of data *sources* and *sink(s)*. A *data source* is a node that produces sensor data to be sent back for processing, and the *data sink* is where all that data gets collected and processed (typically, a base station node). In order to form routes between the sources and sinks, the base station node initiates sensing and data collection phases in the network by flooding out a domain creation/selection message. Once the domain has been formed, data collection and retrieval begins.

The routing process takes place simultaneously with domain formation. As each node floods out the domain command, it also piggybacks information about itself on the packet, such as its distance from the base station. Each of a node's neighbors receives this command and processes meta-information about the source. Each node then attempts to construct the shortest multi-hop path to the base station through other nodes and uses this path to send its data to the sink. We term approach *active routing*.

Active routing is advantageous in networks that have a many-to-one communication pattern, i.e. all communication in the network is only between the data sink and the various data sources, but may not be suitable for other communication patterns.

### 2.3 KUASAR phases and algorithms

The protocol consists of two phases, *Dissemination* and *Collection*. Dissemination employs the flooding

of command messages to each node in the sensor web by the base sink. The command message serves two purposes.

1. To convey the command and parameters over to the sensor devices, and
2. To initiate an active routing process that determines the best path to the sink from each node.

Nodes advertise their distance from the sink as part of the routing information piggybacked on the command. Neighbors receive these messages and make decisions as to which is the best neighbor to relay data. This process allows each node to sense the topology of the network; information about their neighbors; and their own distances from the sink without having to initiate specific routing procedures with each neighbor.

Neighbor information is stored in each nodes *neighbor table*. The neighbor table is a tuple of the form  $\{[N_1, d_1], [N_2, d_2], \dots, [N_n, d_n]\}$ , where  $N_n$  indicates a neighbor's identifier and  $d_n$  indicates the cost of sending data to the sink using  $N_n$  as the next hop. A node's neighbor table thus contains a list of neighbors, which the node can use to relay data back to the sink.

The next-hops and distances calculated are advertised to the next set of nodes further away from the base by way of similar flooding. The dissemination phase ends when all nodes have received commands and have established routes to the sink. Fig. 2 presents the basic dissemination algorithm in pseudo code. When a command message is received from a neighbor, information is entered into the neighbor table and the command is flooded into the network after a random wait period. Additionally any packets received after the broadcast of the route are not forwarded. This means each node forwards a command message and associated routing information just once, ensuring that the dissemination phase converges very quickly.

```

If a command message is received{
    Increment distance to reflect current hop
    Insert route in neighbor table and calculate
    new next hop
    If the route has not yet been flooded {
        Set / Reset the broadcast-timer;
        If broadcast-timer
        expires, broadcast the
        route
    }
}

```

Fig. 2. Pseudocode of Dissemination Algorithm

Collection is the phase in which the sensor devices channel data back to the sink in the paths that have been chosen during the routing process. The collection phase at each node starts off as soon as the command is processed and need not wait until dissemination procedures have been completed

across the network. When the nodes have information about the next hops, they initiate transmission of their sensor data back to the sink. *Acknowledgement* messages are returned for each received message. Ack watchdog timers are implemented to ensure a sending node does not wait for acknowledgement messages from failed nodes. Receive packets are also buffered until acknowledged. This allows a node to retransmit a packet to a new node if the current next hop dies. The size of the buffer also serves as a measure of the congestion at the node. When a node dies, it cannot relay messages and will not send acknowledgements back to a sender node. When the ack watchdog timer expires on the sender node, it reverts to a different node as its next hop. Message formats are very simple containing only the source identifier information and the data to be transmitted. No additional information is required. Fig. 3 provides the pseudo code for the basic collection phase.

```

If data from sensor is ready or data packet arrives {
    Transmit to first node in the neighbor table
    Set/Reset acknowledgement-timer
}
else {
    wait for data to be ready or data
    packet to arrive
}

If acknowledgement-timer expires {
    Remove the first node from the table
    Calculate new next-hop
    If table is not empty {
        Transmit to nearest node in the new table
    }
    go to sleep
}

```

Fig. 3. Pseudo code description of collection algorithm

#### 2.4 Example Scenario

Consider the set of sensor nodes and base station configured as shown in fig. 4. When the base station has a requirement for sensor information, it broadcasts a command to the network as shown in fig. 5. The command floods through the network first to node 1 and node 2. These nodes wait a random amount of time for other nodes to announce themselves before flooding the packet. Node 1 and node 2 then set their next hops to be the base station, as shown in fig. 6, and then flood the packet through the remainder of the network. This flood also reaches the base station where it is known that node 1 and node 2 are nearest neighbors.

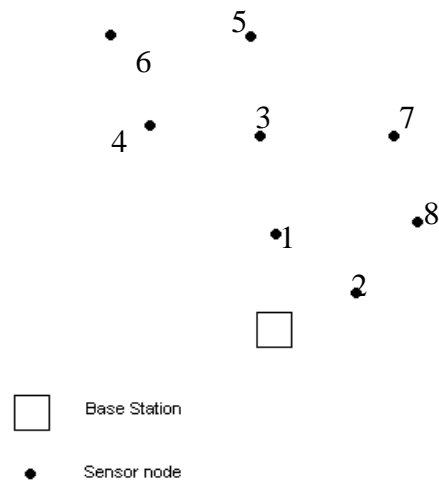


Fig. 4. An example sensor web

The next broadcast wave is picked up by node 3 and node 8. These nodes set their next hops to node 1 and node 2 respectively. Note that node 1 and node 2 also receive each other's broadcast packet but recognize that they are in each other's neighborhood. Node 4, node 5 and node 7 pick up the third wave from node 3 and node 8. Node 1 and node 2 also receive broadcasts from node 3 and node 8 and both record's them as neighbors. The connectivity network shown in fig. 7 is finally developed. The neighbor tables of each node are listed in figs. 8.1-8.4.

It is easily recognized that the hop-count numbers for each of the neighbors of the nodes other than the next-hop do not accurately reflect the true state of the network, in that some nodes are wrongly recognized as potential next-hops. This is dealt with in later sections. Every node that completes the routing process immediately begins collecting and transmitting data to the base station.

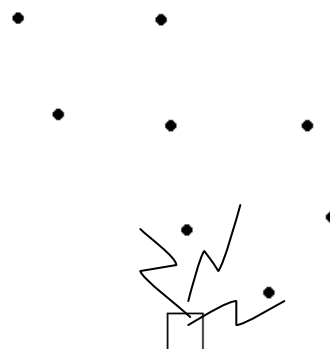


Fig. 5. Dissemination begins. Base station broadcasts command packet

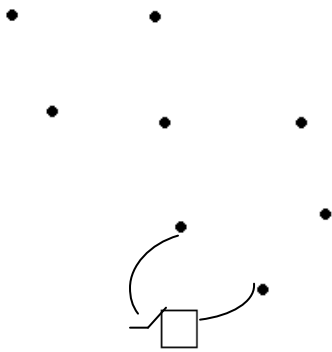


Fig. 6. Nodes start forming shortest paths to the base station.

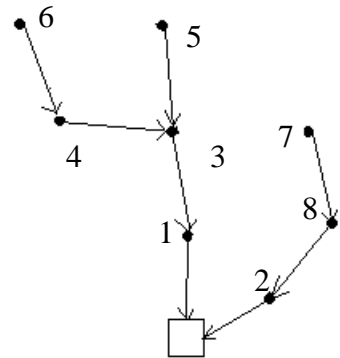


Fig. 7. The sensor web after the dissemination phase

0	1
2	2

0	1
1	2

Fig. 8.1. Tables at Node 1 and Node 2

1	2
4	4
5	4
7	4

3	3
6	5

Fig. 8.2. Tables at Node 3 and Node 4

3	3
6	5

4	4
5	4

Fig. 8.3. Tables at Node 5 and Node 6

8	3
3	3

2	2
7	4

Fig. 8.4. Tables at Node 7 and Node 8

Consider a fault occurs that results in the loss of the link between node 1 and the base station. This implies that the base station cannot receive anything when node 1 transmits a packet. In this case node 1 does not receive an acknowledgement from the base station. This prompts node 1 to search for an alternative path in its routing table. It finds node 2 is the next best (in fact, the only) option, and makes node 2 its next hop. This is a simple scenario that illustrates how multiple paths are formed. However, as mentioned earlier, this can result in a subtle problem with this mechanism that will be discussed in section 3. Section 3 also outlines how KUASAR reacts to changes in topology, and how loop-free multi-paths are achieved.

### 3. PROPERTIES OF ACTIVE ROUTING

Active routing guarantees a shortest multi-hop path from the sources to sink. This can be proved as follows. The KUASAR algorithm is a greedy algorithm that always chooses the node that advertises the shortest possible path to the base station. Let the network be represented as a graph  $G(V,E)$ , where  $V$  is the set of nodes that contain among others, the sink  $s$  and a node,  $x$ .  $E$  is the set of weighted edges, and  $w_t(E)$  is the weight of  $E$ . In KUASAR, the weight of an edge is the number of hops to the sink including that edge.

Suppose KUASAR identified a path  $P$  from node  $x$  to the sink  $s$ . Suppose also that it is not the shortest path. This implies that there is a different path  $P'$  that is shorter than  $P$ . For  $P$  and  $P'$  to be different, then there must be a unique edge  $E = (V_1, V_2)$  in  $P$  not in  $P'$ . That implies there must also be an edge  $E' = (V_1, V_2)$  in  $P'$ , such that  $w_t(E') < w_t(E)$ . But, the next edge that KUASAR selects at every stage is always of the shortest distance to the sink. This means either  $E$  and  $E'$  are the same edge, or  $w_t(E) = w_t(E')$ , which is an equally acceptable case (two distinct neighbors of  $E$ , both have the same distances to the sink, so any one can be chosen).

#### 3.1 Reaction to topological change

Topological changes can be caused due to node deaths, node additions, or node movement. In all three cases the routing pattern is disrupted and route loops may occur. Most routing protocols tackle topological change by well-timed updates to the routing tables to reflect these changes. These updates can be either of two types – temporal or triggered (Broch *et. al.*, 1998). KUASAR follows a path that is neither as proactive as a temporal update nor as passive as a triggered update. As mentioned earlier, KUASAR initiates the routing mechanism but lets nodes themselves discover paths to the sink. The shortest path is always selected for forwarding whereas the other paths are treated as backup paths to be used in case of intermediate node failures. If a new node enters an already existing routing grid, that



## REFERENCES

- Broch, J., Maltz, D., Johnson, D., Hu, Y., and Jetcheva, J. (1998). "A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols". In Proceedings of the Fourth Annual ACM/IEEE International Conference on Mobile computing and Networking, ACM, Dallas, TX
- Chen, Y., Liu, A., Zhang L. (2001). "A scalable solution to minimum cost forwarding in large sensor networks", Proceedings of the Tenth International Conference on Computer Communications and Networks, pp. 304-309
- Culler, D., Hill, J., Buonadonna, P., Szewczyk, R., and Woo, A., (2001). "A Network-Centric Approach to Embedded Software for Tiny Devices", Proceedings of the First International Workshop, EMSOFT 2001, Tahoe City, Ca.
- Heinzelman, W., Chandrakasan, A., and Balakrishnan, H., (2000). "Energy-Efficient Communication Protocols for Wireless Micro sensor Networks", Proc. Hawaaiian Intl Conf. on Systems Science
- Hill, J., Szewczyk, R., Woo, A., Hollar, S., Culler, D., Pister, K., (2000). "System Architecture Directions for Networked Sensors", Architectural Support for Programming Languages and Operating Systems
- Intanagonwiwat, C., Govindan, R., and Estrin, D., (2000). "Directed Diffusion: A Scalable and robust Communications Paradigm for Sensor Networks", ACM MOBICOM'00
- Kulik, J., Heinzelman, W., Balakrishnan, H., (1999). "Negotiation-based protocols for disseminating information in wireless sensor networks". ACM/IEEE International Conference on Mobile Computing and Networking
- Rentala, P., Musunuri, R., Gandham, S., Saxena, U., (1999). "Survey on sensor Networks", Technical Report Department of Computer Science, University of Texas, Dallas
- Royer, E., Toh, C-K., (1999). "A Review of Current Routing Protocols for Ad-Hoc Mobile Wireless Networks", IEEE Personal Communications Magazine, pp. 46-55.
- Sohrabi K., Gao, J., Ailawadhi, G., J. Pottie, (2000). "Protocols for self organization of a wireless sensor networks". IEEE Personal Communications, Vol. 7, Issue 5, pages 16-27, 2000.
- Wang A, Chandrakasan A, (2001). "Energy efficient system partitioning for distributed sensor networks", IEEE International Conference on Acoustics, Speech and Signal Processing, Vol. 2. Pp 905-908