

Constructing Wireless Sensor Networks via Effective Topology Maintenance and Querying

Sebnem Baydere, Mesut Ali Ergin and Ozlem Durmaz

Department of Computer Engineering
Yeditepe University, 34755, Istanbul, Turkey
{sbaydere, ergin, odurmaz}@cse.yeditepe.edu.tr

Abstract—The ability of properly covering the terrain under investigation and collecting measurements from redundantly sensed portions of the terrain are two important objectives for monitoring applications using wireless sensor networks. Here, a new architectural view of information retrieval for XML compliant environmental monitoring applications is introduced. The sensor network is constructed to satisfy the requirements of monitoring applications and maintained as long as the application has queries to be run on the sensor nodes. Mobile clients of the architecture (drivers), such as human beings or autonomous robots, navigate within the sensing environment and build up sensor node trees in order to effectively disseminate queries and collect the results. In the paper, viable methods are proposed for *query service* binding, query driven sensor network *topology construction* and *end-to-end event delivery* on dynamically maintained return paths. Simulation results considering the reliability and coverage performance of the proposed approach are provided to evaluate the new schemes.

Keywords— Environmental monitoring, Sensor networks, Routing, Network reliability

I. INTRODUCTION

Wireless sensor networks (WSN) became a leading research and development area arose from the field of ad hoc networking research. Increased research activity is due to the exciting and convincing reasons provided by the potential for significant monitoring applications on various different subjects such as heat, humidity, light or several artificial structures. Countless environmental sensing applications from pollution estimation to fire-fighting will be in use once these networks are deployed in wide scale.

Embedded networked environmental sensing systems use distributed micro-sensors and benefit from low power

This work is partially supported by The Scientific and Technical Research Council of Turkey under grant 103E001/EEEAG-AY-44.

wireless communication to collect and disseminate environmental data. Each sensor node is designed to monitor the environment for events specified in the queries, created by the applications. These events may be specified during deployment. For such applications, sensor nodes are assumed to be densely deployed within a geographical area in an ad hoc fashion and operate in an unattended and untethered mode. Collaboration among hundreds to several thousands of limited computation, communication and sensing capable sensors is needed for information retrieval in the area of interest. Several design challenges exist for intelligent query processing besides networking of such devices. Thus, in this paper, we focus on query dissemination and information retrieval on a hierarchical network architecture. We propose algorithmic solutions for the following two problems:

- How queries produced by web-based monitoring applications can trigger adaptive construction of a chain of collaborative tree networks?
- How query dissemination and data routing on dynamically maintained return paths can be achieved?

Main objective of introduced methods is to effectively cover the area under investigation and return the maximum possible amount of valuable information to monitoring application. Our proposal for sensor networks aims to address communication needs of monitoring applications that are composed of randomly deployed stationary sensors which compile a measured value correlated to their current positions. The terrain of the sensor deployment area is assumed to be suitable for navigation by means of some mobile units, which form the information retrieval backbone of the overall monitoring application. This backbone relies on a service-aware wireless ad hoc network protocol called SeMA [1]. The mobile units of the backbone are either wireless

equipment carrying livings or autonomous robots as seen in multi robot exploration studies [2].

An illustrative example of sensor network usage in such monitoring applications can be given as follows. We may need to detect the amount of hazardous chemicals that have spread over an urban environment, because of a leak during transportation or explosion of an on-site reactor. Wireless communication capable, chemical material density measuring sensors are deployed over the polluted area in an ad hoc fashion. The area under the pollution is very large and communicating with each and every sensor from a central administration point is practically impossible. In this scenario, hazardous material density is the target to be monitored from a safe distance. For the information retrieval, autonomous robots are released to the area, where each will initiate a temporal sensor tree topology to disseminate and gather results of a query created by the monitoring application. The monitoring application running host and the autonomous robots (i.e., *drivers*) form a wireless backbone network, with comparably longer transmission and reception capable radios and better energy capacity holding batteries. Drivers of the backbone network relay each other to announce the queries of monitoring service or to collect results of the queries back to monitoring application. Queries trigger the sensors to let them participate in reporting their hazardous material density measurements.

Remainder of the paper is organized as follows. Section II discusses related work on query dissemination, processing and data routing. Section III explains components of the proposed hierarchical network architecture. Section IV describes the sensor model and the phases of the topology construction algorithm. Section V gives quantitative arguments for network connectivity and area coverage capabilities of sensor networks of concern. Finally, concluding remarks are provided in Section VI.

II. RELATED WORK

Trying to network a large number of low power sensor nodes is a challenging problem that has recently been the focus of a growing research effort [3], [4]. Particularly data routing, node addressing and support for different classes of services are the primary issues tackled at the network layer [5]. Traditional ad hoc networking protocols discussed in [6], [7] are generally not resource-aware or resource-adaptive. On the other hand, conventional data dissemination approaches like flooding and gossiping waste valuable communication and energy resources by sending redundant information throughout the network. Therefore, these schemes have been difficult

to adapt, and as a result many new algorithms have been developed [8], [9], [10].

An important design goal for a routing algorithm proposed for sensor networks is to enhance the system lifetime, dependent on the limited energy of sensors. Different routing protocols propose solutions that use energy efficiently by different approaches.

LEACH (Low Energy Adaptive Clustering Hierarchy) [4] is designed for sensor networks where an end-user wants to monitor the environment remotely. In such a network, the data from the individual nodes must be sent to a central base station, often located far from the sensor network. The end-user can access data through this station. It includes distributed cluster formation, local processing to reduce global communication, and randomized rotation of the cluster-heads. The data are collected in a centralized manner (via cluster heads) and periodically. This can be said to be most appropriate only for constant monitoring. Periodic data transmission may be unnecessary for some applications (i.e., event based monitoring).

Directed Diffusion [11], is a destination initiated reactive protocol working well for queries like '*send me temperature data in a particular area*' and responses returned to such queries. A controller requests data by sending an interest flooding over the whole network. Variations of the protocol with distinguished high and low rate paths are also proposed. This protocol may not be suitable for one-time queries since the sink refreshes and reinforces the interest from the source.

SPIN [12] is a family of protocols used to disseminate information in a WSN efficiently. It solves shortcomings of conventional approaches using data negotiation and resource-adaptive algorithms. Nodes running SPIN assign a high-level name to their data, called meta-data, and perform meta-data negotiations before any data are transmitted. This assures that there is no redundant data sent throughout the network. In addition, SPIN has access to the current energy level of the node and adapts the protocol according to the amount of the remaining energy. While negotiating the data, SPIN distributes information all over the network, even a user does not request any data. This generates an overhead for some query forms.

Although resource-awareness is considered in the current routing protocols, degree of scalability and robustness requirements of sensor networks (established to run long-lived operations in dense sensing terrains) are still in question. Additionally, leveraging topological location, independent of any application [13] and

heavy use of neighborhood information in distributed or centralized routing algorithms over the established coordinate systems make proposed network solutions computationally expensive in terms of the messages exchanged [7], [14], [15]. Energy consuming functions of such coordinate based network layer protocols trade off protocol performance and battery life-time.

Dependence on topological location also exists in solutions proposed for energy aware routing with synchronized sleep and wake-ups at the MAC level [16] or connectivity optimization in low energy networks [5], [17], [18], [19].

Considerable energy has been devoted to routing protocols for managing underlying sensor networks and research efforts on query processing environments for data dissemination has been emerging recently. TinyDB Query Processing System [20] provides a high-level, declarative language for specifying queries accessed from a connected PC for query construction. Assuming TinyOS [21] components are installed onto each mote class sensor node in the sensor network, TinyDB collects data from motes in the environment, filters it, aggregates and routes it out to a PC for a given an SQL-like query specifying user's data interests. In TinyDB, a single infinitely-long hypothetical table, called *sensors*, is implicitly queried. Another data extraction mechanism, ACQUIRE, is proposed in [22] for sensor network querying where active queries are injected into the network with triggered local updates.

Although our goal is the same as TinyDB in principal (providing a multi-tiered query processing environment for sensors), there are some architectural differences in handling the above mentioned problems. Our architecture supports XML [23] format for service definitions. Thus, underlying network infrastructure is completely transparent to web-based monitoring applications conforming XML standards. XQuery [24] language is used within the query that is disseminated on the ad hoc backbone. The architecture also supports concurrently constructed rooted sensor trees running the same or different queries. This facilitates scalability in sensing terrains. Traditionally, network layer routing and query dissemination problems are handled in separate layers of the protocol stacks. In a TinyDB network, root node periodically announces itself in control packets so that sensors can update their parents and maintain the routes towards the root. Queries are disseminated and maintained in data packets. In our scheme, tree network is formed in parallel to query dissemination. Route maintenance on initially constructed return paths

is done during data delivery to the root, with a dynamic path switching algorithm. In ACQUIRE, for example, returning the result for a query is done by relaying the partially collected answers to a random node within a limited distance. For queries like '*return temperature reading from all sensors*', top-down tree maintenance with periodic control packets can be justified. However for queries returning rare events such as '*return sensor readings if temperature > 50°C for the next one hour*', overwhelming control packet processing should be avoided for energy saving reasons. Such a query may not generate data traffic towards the root at all, if the event does not occur in the network. Additionally, as a part of the route maintenance algorithm, our protocol supports end-to-end reliable event delivery on the network.

III. SYSTEM COMPONENTS

In this section, we explain the components of the service-aware backbone network and the underlying adaptively formed query driven sensor network architecture. Main characteristics of the interacting components are given in four models; *Service Model* describes a monitoring application, running on the service-aware network. *Client Model* describes cluster heads (drivers). They are mobile nodes capable of acting as an intermediary between a distinct set of randomly deployed sensors and an upper layer monitoring service application. *Network Model* gives the hierarchical interaction between system components and *Sensor Model* states the characteristics of the sensing devices and sensed data.

A. Service Model

Monitoring applications are developed using features from a service-aware ad hoc network protocol stack, named SeMA [1]. It is capable of operating on well known wireless link protocols (i.e., IEEE 802.11). SeMA is a general purpose wireless ad hoc network protocol, which supports representation and announcement of services in XML-formed attribute-value pairs for binding. Mobile clients, navigating in the monitoring area of interest, can *fetch* and *bind* to services announced in the network dynamically without any need for static configuration or central directory services. SeMA has been developed as a service-centric protocol stack with high varying topologies in mind. For this reason, routes and sessions are not maintained for destination nodes but for service instances.

The query which will be carrying the objective of the monitoring application travels through the SeMA

```

<service name="monitoringApplication">
  <keyword attribute="validUntil">20031128193044EEST</keyword>
  <keyword attribute="queryPredicate">
    sensor:read[value>100 and order(value)<5]
  </keyword>
  <keyword attribute="resultFunction">fn:concat</keyword>
</service>

```

Fig. 1

XML DEFINITION OF A MONITORING SERVICE

backbone in the announcement packet of the monitoring service. This service definition homes many parameters such as a query validity time window, an XQuery predicate and function. XQuery predicate is used to extract the readings that interest the monitoring application. Then, if specified in the service, these readings may be processed via the given resultFunction. The actual query result to be submitted is the returned data from this XQuery function. XQuery specifies more than 200 functions (including functions in SQL) that include numerical processing, data aggregation (*sum*, *avg*, *min*, *max*), string operations (*string-join*, *starts-with*, *ends-with*), pattern matching (*matches*, *replace*, *tokenize*) etc. and more. By using XQuery in value fetching and processing, sensor querying process conforms to XML standard, from monitoring application down to the sensor nodes. Clients collect query responses and return compiled results to the monitoring service in data packets. A correlated diameter of the event instance is also returned. A sample monitoring service definition is given in Fig. 1. In this example, service attributes are defined as:

- **validUntil:** Query validity time.
- **queryPredicate:** An XQuery predicate that will filter out the desired sequence of values from the sensor readings. To be more specific, the given predicate instance will fetch the last five readings that carry numerical values greater than 100.
- **resultFunction:** A built-in or custom XQuery function that will be applied to the resulting sequence of the given predicate. What the function returns will be sent to the client sensor node driving the tree. For the example instance, the sequences that are found by the predicate will be concatenated to form a compact resulting string of values.

B. Client Model

Clients are mobile nodes modelled with two wireless network interfaces; Communication interface which runs

SeMA ad hoc backbone protocol, is used to fetch monitoring service instances for query attributes. All data required to drive a sensor network topology (such as the query, query time boundaries, runtime characteristics, geographic region boundaries etc.) are encapsulated in the service attributes. Upon fetching a service, mobile client pauses its movement, activates the sensor *driver* process and goes to sleep. The driver process initiates a rooted tree network among a subset of all sensors in the region via the node's second wireless network interface. This is a short range radio capable of communicating with sensor node radios in the same transmission range. This interface is used to disseminate the query in a controlled way. During *topology construction* and *query running* phases, the driver node is assumed to be fixed; not moving. Upon collecting data generated by the events matching queries on the sensor network, the driver wakes up the client and an implicit SeMA session with the monitoring service is established for transmitting collected query responses. SeMA is a best effort packet delivery protocol, which supports duplicate detection and sequenced delivery without re-transmission at the network level. During a session, route losses on the backbone (possibly caused by node mobility or an exhausted battery) are handled by modifying the source route in the SeMA data packets, with the alternative routes for the same service announcement. The rest of the backbone architecture is discussed in Section III-C.

In summary, underlying network infrastructure enables monitoring services to be discovered dynamically via multiple mobile clients. These clients adaptively initiate non-overlapping sensor trees in geographically overlapping sensing areas. Topologies driven by the same query represents the whole sensing terrain. It is also possible to manipulate the topological changes by modifying the service attributes and re-announcing the monitoring service. Without any manual intervention, different queries can be disseminated in the same geographic area over non-intersecting time periods.

In our model, we assume the existence of at least one monitoring service instance and at least one active mobile client accessing this service.

C. Network Model

Upon activating driver process, sensor *driver* node constructs a small size (a few bytes) setup message and broadcasts in its transmission range. Setup packet format is given in Fig. 2. The setup message, carrying query parameters, initiates a *depth n* rooted tree network topology as described in Section IV. Sensors match query

attributes to generate responses and the network routes them back to the root node. Geographic accuracy of the results are correlated with the distance of the sensor from the driver node. The maximum distance is the number of hops n multiplied by the radio transmission range k of each sensor on the return path. Accuracy diameter is illustrated in Fig. 3. Multiple clients can concurrently initiate non-overlapping tree topologies in possibly overlapping physical regions. Each tree is identified uniquely by its root node which is a distinct sensor driver mapped to a virtual or physical region identifier for query processing. Each sensor is allowed to join one tree at a time to process the query disseminated by its root node. Therefore, a group of sensors in the same transmission range may become nodes of different trees. As sensing diameters may overlap, non-overlapping virtual tree regions can map to the same physical region resulting in multiple nodes in different trees generating the same response to the query. This may cause redundant responses for the same phenomenon to be returned to the application via different roots. This feature will later be used to optimize the reliability defined as the ratio of long term area coverage with reduced number of nodes

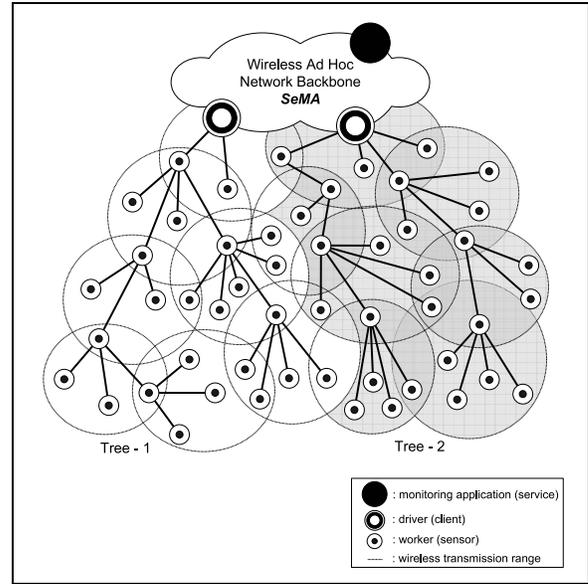


Fig. 4
SYSTEM INFRASTRUCTURE

| Packet Type | Region ID | Node ID | Flooding Degree | Hop Count | Query Timeout | Acknowledge Flag | Concatenate Flag | Query Related Parameters |
|-------------|-----------|---------|-----------------|-----------|---------------|------------------|------------------|--------------------------|
| 2 bit | 16 bit | 16 bit | 8 bit | 8 bit | 13 bit | 1 bit | 1 bit | 23 - 148 bit |

Fig. 2
SETUP MESSAGE FORMAT

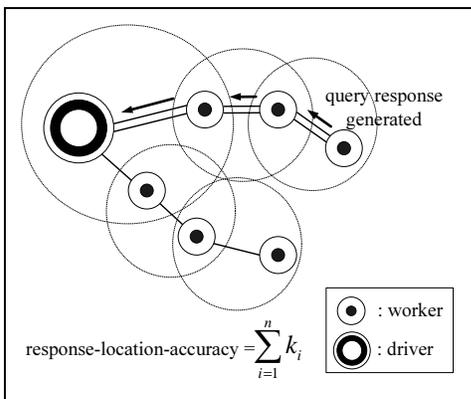


Fig. 3
ACCURACY DIAMETER OF A 3-HOP SENSOR NETWORK

due to battery run out. For the time being, each physical region is uniquely identified and mapped to a single driver node which possibly is a part of a location tracking system. Filtering criteria for multiple replies returned to the application from overlapping physical regions may be defined in the service attributes. Network infrastructure as shown in Fig. 4 is a combination of adaptively formed rooted trees where roots are the clients of the service-aware backbone. The overall architecture is illustrated in Fig. 5 as the interaction of protocol stacks on each system component.

D. Sensor Model

Sensor nodes of the embedded network are called *workers*. They are assumed to be operating with independent fail probabilities in untethered and unattended mode. Their lifetime is limited with the battery life where their sensing units, processors and radios withdraw power independently, if they have not been switched off. *Workers* simply wait in idle state to receive a setup message and decide whether to participate in the query network or not based on their remaining battery level. If a worker decides to participate, then it activates its sensing unit and switches to query processing mode.

Worker sensors do not necessarily act on a coordinate system with unique node addressing or knowledge of neighborhood assumptions. They are assumed to have

two interfaces. *Communication interface* is a low bit rate, short range packet radio transceiver (with a few meters transmission range), operating a broadcast MAC protocol. *Sensing interface* processes sensed data for query execution. For simplicity, it is assumed that sensing and radio transmission ranges of a sensor node may overlap and sensors are fixed nodes with a limited lifetime. During its lifetime, each worker moves between three states; *idle*, *wake-up* and *active*. In active state, sensor may be driven to sense movement or to read environmental data such as temperature or humidity. Sensor networks can be organized for the purpose of tracking a (moving) target between virtually non-overlapping regions or simply sensing environmental data with the given location accuracy.

IV. TOPOLOGY CONSTRUCTION

Workers are assumed to be randomly deployed sensors in a geographical area sharing a broadcast communication medium with an unknown number of workers. Three types of network packets are defined: setup, data and acknowledgment. Nodes either process a setup packet or a data/acknowledgment packet but not both during the same time period. As MAC level links are not necessarily established, network packets do not contain source and destination addresses. Sensors are assumed to have limited energy, data processing capacity and broadcast transmission capability. Knowledge of neighborhood, their power capacities or dissemination of additional control messages are not required in the network protocol. Over such a simple communication medium an n-hop tree

topology is constructed with the implicit intention of event delivery. The data-centric routing protocol optimizes this delivery on virtually established links with an adaptive path switching algorithm.

A. Initialization Phase

In this section, we explain the initialization algorithms for the driver and worker nodes. At the end of this phase, those sensors joining a tree, activate their sensing units and run the disseminated query for the query validity time period as defined in the attributes.

Drivers are activated by client applications with fetched query parameters to run the following topology construction algorithm:

- 1) Extract query related parameters from the service attributes. These attributes contain the query and query timeout boundary values.
- 2) Generate global parameters; *flooding degree* (fd): effective diameter of the tree in terms of maximum hop count to the driver, *region identifier* (reg_id): unique tree id derived as a function of the driver's geographic location, and *node identifier* (node_id): locally generated random identifier for the node.
- 3) Construct setup message and broadcast.

Region identifier in the message is used to identify the tree. This id may be mapped to a virtual region by the monitoring service. Virtual reg_id may be passed to the driver node as a service attribute or driver's coordinates can be mapped to a virtual region_id by the service. Global information is passed to sensors in setup message. Driver initialization algorithm is given in Fig. 6.

Sensors are in idle state at deployment; i.e. listening network interface in low duty cycles with their sensing

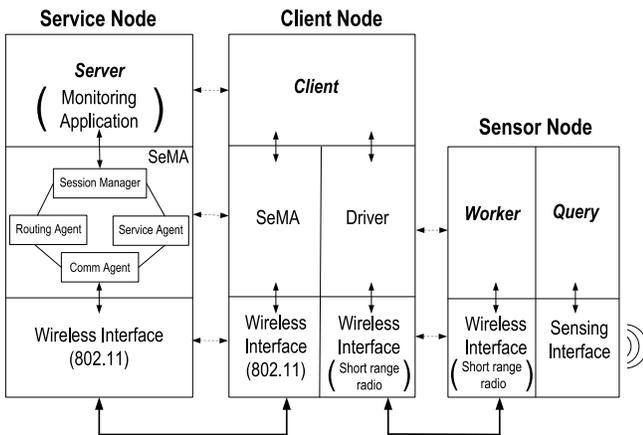


Fig. 5

OVERALL NETWORK ARCHITECTURE

```

Fetch(monitoring_service);
Set node_type = sensor_driver;
If(got GPS)
    reg_id = GPS.location;
Else
    reg_id = monitoring_service.reg_id;
Set query = monitoring_service.query;
Set timeout = monitoring_service.timeout;
Set node_id = f(reg_id);
Initialize net_params (fd=n, hc=1);
Set message.type = setup;
Broadcast message(message.type, query, timeout,
    reg_id, local_node.node_id, net_params);
Sleep Until message.type = reply arrives;

```

Fig. 6

DRIVER INITIALIZATION ALGORITHM

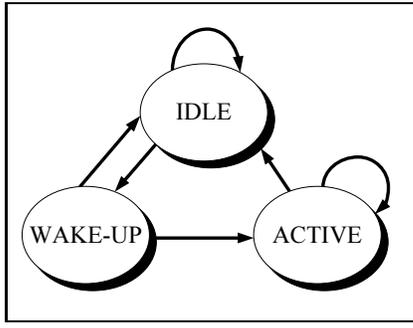


Fig. 7

STATE TRANSITION DIAGRAM FOR WORKERS

units switched off, as suggested in [5]. They move to energy consuming mode during topology construction phase and move back to energy saving mode again when the query is expired. In processing a setup message if a sensor decides to become a node of this query network, it activate its sensing unit. All succeeding setup messages will be ignored by this node until the end of the query running phase (i.e., expiration time). Workers insert *reg_id* and *node_id* extracted from the setup message in all data packets they transmit. This way, a logical return path is established with the node sending the setup message and its immediate receivers. In processing the setup message, the worker does the following:

- 1) Decrement flooding degree (fd) by one. This field is used as a TTL flag.
- 2) If TTL degree reaches zero then stop flooding and switch to query running phase.
- 3) If TTL degree is greater than zero then generate a random local *node_id* and replace the *node_id* field in the setup message with this locally generated *node_id* and broadcast the setup message.

Each network is constructed to run a single query. Construction cycle is recursively repeated over a set of sensors and succeed on a subset of it. For each node the cycle is repeated as long as the sensor has enough battery life to participate in the next cycle. During its lifetime, a worker moves between three states: idle, wake up and active as shown in Fig. 7.

- **Idle:** Initial state at deployment. The node is idle listening its network interface with sensing function off. Energy consumption is very low.
- **Wake up:** Moved when a setup message is received. The node decides whether to participate in the topology or not. If it decides to participate then saves the query, runs the construction algorithm and

activates sensing unit. If it decides not to participate then it returns back to idle state.

- **Active:** Moved when the node's sensing unit is activated. Node runs the query, generate a local response if necessary or act as an intermediate node on a response path.

Initialization algorithm for workers is given in Fig. 8. At the end of this phase, all participating sensors switch to sensing mode and become capable of detecting an event as dictated in the query. Event detection causes a worker to insert the extracted *return_node_id* in a data packet and transmit it. As this identifier was generated by the parent node in the sender's immediate transmission range, only this node takes the responsibility of forwarding the event towards the root upward in the tree by incrementing the hop counter and changing the *return_node_id* field in the packet. Other nodes do not process the packet unless path switching due to link lost is required. In that case, data path is dynamically switched as explained in the Section IV-B.

B. Query Running Phase

In this section, we explain the operations performed by the workers in energy consuming mode for the duration of query timeout period. In this phase, workers periodically read local data and process data/acknowledgment packets arriving at their communication interface. They compare locally sensed data with the query threshold and generate a response if necessary. Additionally, in

```

Set sensing = off; state = Idle;
Receive(message);
If(message.type = setup)
  Calculate remaining energy;
  If(decide not to participate)
    return;
  Else
    Set sensing = on; state = Wakeup;
    Set return_node_id = message.node_id;
    Set local_diameter = hc; (# of hops to the root)
    Set timer = message.timeout;
    Set query = message.query;
    Set net_params (hc = message.hc++,
                  fd = message.fd);
  If(fd > hc)
    Generate local_node.node_id = f();
    Set message.type = setup;
    Broadcast message(message.type, query, timeout,
                    reg_id, local_node.node_id, net_params);
    Set state = Active;
  
```

Fig. 8

WORKER INITIALIZATION ALGORITHM

receiving a data packet, workers first compare the `reg_id` to detect if the packet belongs to this network or not. If so, they compare the `return_node_id` with their locally generated `node_id` to see if they are on the return path for this data packet or not. For valid paths, prior to forwarding, workers update the `return_node_id` field and the hop counter in the data packet. Query running algorithm for the best effort delivery network is given in Fig. 9.

If end-to-end reliability between the event and the driver node is required then *QoS* flag is set in the setup message during the initialization phase. All data packets generated in this network carry this flag. With flag unset, the network is configured to provide a best effort delivery of replies. With flag set, the network provides a reliable event delivery by dynamically changing the next hop sender to be the new forwarding node in the return path. Therefore, end-to-end transmission reliability is provided between the last node on the return path which has successfully received the event and the driver. The original return path is dynamically changed if the next node on the path disappears. Dynamic path switching algorithm works as follows: In processing a response packet, besides updating the `return_node_id` and forwarding the data, the worker generates a backward control packet. As end-to-end transmission is required between the event and the driver, this control packet cuts the logical link with the previous node on the route. Therefore, each

node on the return path takes the responsibility of being the event sender until the control packet from the upward node on the path is received. If no acknowledgment is received within the timeout period then the node clears the `return_node_id` field and re-transmits the data packet. Those nodes in the immediate transmission range of the sender, receive the packet with empty `return_node_id`. Each receiving node compares its local diameter with the sender's diameter. If local diameter is smaller than the sender's diameter then the local node takes the role of being the event sender and behaves as if the data packet has arrived from its own child. All receiving nodes with smaller diameter wait a random amount of time (penalized with a diameter correlation factor) to hear an acknowledgment before applying the take-over algorithm. In other words, response packet changes its return path on the tree dynamically. Steps of the dynamic path switching algorithm are shown in Fig. 10. In Fig. 10(a), worker 4 generates an event over the path towards the root. This event is received by worker 3 successfully and forwarded to worker 2 which is down. Worker 3 detects it and re-transmits the data with empty `return_node_id` field. Next step which is illustrated in Fig. 10(b) shows the path switching. Workers 5, 6, 7, and 8 compare their local diameters with the sender's diameter. Worker 8 decides to take over the data on its data path as its diameter is smaller than sender's diameter. Other nodes hearing path switching, worker 6 and 7 in the example, also change their `return_node_id` values to be the worker 8 as shown in Fig. 10(c).

Network survivability analysis given in Section V assumes best effort delivery of replies for simplicity. The positive effect of dynamic path switching on the reliability is ignored. Therefore, the results presented in the next section illustrate the worst case performance of the proposed model.

During query running phase, sensor driver is blocked waiting in receive mode for responses. Each arriving data packet contains the event location accuracy diameter as described before. *Response Compilation* attribute of the query dictates the way that the client compiles the results and returns to the monitoring application. (e.g., *all* meaning concatenate and submit all the results, *avg* meaning take an arithmetic average etc.)

V. TREE TOPOLOGY ANALYSIS BY SIMULATION

The overall architecture is comprised of a collection of n-hop sensor networks tied together by a backbone of sensor drivers. Each sensor network can be perceived as a reverse multicast tree with information routed to the

```

While(timer not expired)
  SENSING INTERFACE
  If(sensing = on)
    Read (value);
    If(value matches the query)
      Set hc = 0; message.type = reply
      Broadcast message(message.type, value,
        local_node.return_node_id, hc);
      Calculate energy_consumption;
      Set state = Idle; sensing = off;
  RADIO INTERFACE
  Receive (data.message);
  If(message.return_node_id = local_node.node_id &
    reg_id = local_node.reg_id);
    Set hc = hc++;
    Set message.return_node_id =
      local_node.return_node_id;
    Broadcast (message);
End
Return INITIALIZATION PHASE;

```

Fig. 9
QUERY RUNNING ALGORITHM

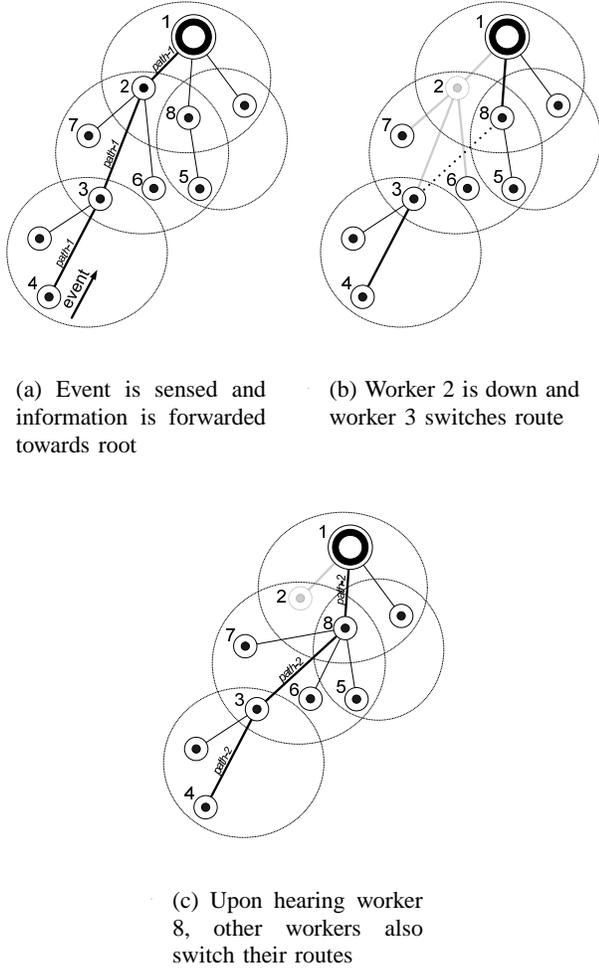


Fig. 10

DYNAMIC PATH SWITCHING

sink node at the root. The tree is randomly constructed on overlapping broadcast communication mediums as explained in Section III. Initially, all sensors in the driver's transmission range establish links with the root. Each tree node recursively creates logical links with the nodes in its transmission range those have not been connected to a tree yet. Since parent selection is random, different trees can be constructed from the same sensor set. In this section, we have analyzed the connectivity of randomly constructed sensor trees for single query under the assumption that query time is discrete and nodes fail with independent probabilities between queries [25]. Then, an exponential battery consumption model is added to the network reliability analysis. Reliability is defined as the survivability; fraction of nodes connected to the tree for each consecutive time limited query. Each consecutive

query results in a tree topology with less (or equal, if no node has exhausted its battery) number of sensor nodes involved, because of the continuous consumption of battery power at sensor nodes. $R(t)$ represents the advance of these consecutive queries for discrete t times. We have tested the construction algorithm up to 10 consecutive queries for different node availability rates representing initial battery levels. For each discrete time unit, battery power is consumed by 10% of its initial value. We also have verified the simulations for a single query with the analytical results at given battery levels and then extended the simulation for the reliability analysis up to 10 queries. Connectivity of regional tree topologies were used to measure the overall connectivity and reliability of the query driven sensor network architecture.

A. Long Term Connectivity

Each node fails with independent probabilities. In best effort delivery mode, at the extreme end, failure of any node in the tree disconnects the sensor network assuming that the nodes on this path detects an event. For simplicity, we have also assumed that the root node r never fails. The availability of a sensor network T , is the probability of full connectivity of the tree. The probability of the tree network being connected, T , is simply the probability that all components are working, which is given as:

$$p_C(T) = \prod_{i=1}^N p_i \prod_{j=1}^M p_j \quad (1)$$

where there are N nodes, and M links.

Sensor tree is a rooted tree where r is the distinguished node, called sensor driver. Although the communication is bidirectional in a broadcast medium, computationally each established link is viewed as directed towards the sensor driver. Each node except r has a unique predecessor, but can have more than one successor. Nodes with no successors are leaves. Total number of nodes having a link in the tree is a subset of the total number of sensors in the field. We calculate the expected number of nodes communicating with r , $E(r)$ by summing $p_C(i)$ over all i values,

$$E(r) = \sum p_C(i) \quad (2)$$

This calculation is $O(Nm)$ where N is the number of nodes and m is the average depth. The expected number of nodes communicating with r is simply the sum of the expected number of nodes communicating with each of the successors of i , multiplied by the probabilities of

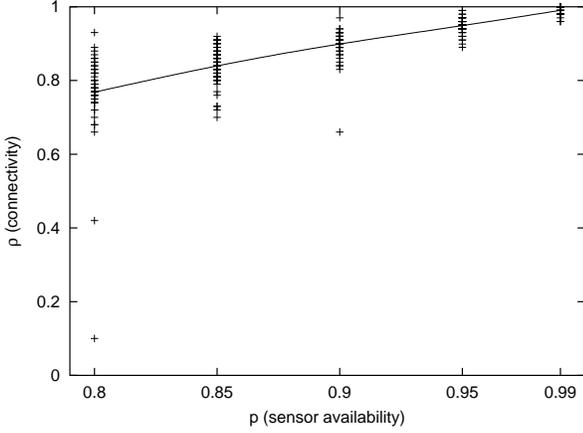


Fig. 11
CONNECTIVITY RATIO FOR SINGLE QUERY

these successors communicating with i , given as,

$$E(i) = p_i + p_i \sum_k p_{jk} E(k) \quad (3)$$

Where $E(k)$ is the expected number of nodes communicating with k , and jk is the link connecting k to its predecessor and sum is taken over nodes which are successors of i . If i is a leaf, (3) becomes $E(i) = p_i$. For simplicity we assumed reliable links in (1) and (2). We have simulated a region with 100 nodes for initial sensor availability rates p ranging between 0.99 and 0.80. We have run the simulations 50 times for each battery level and calculated the fraction of nodes connected to the tree. For a single query, connectivity ratio variance for 50 random trees is found to be in acceptable bounds. The results are illustrated in Fig. 11. Simulation is then extended for 10 consecutive queries to analyze the long term connectivity. Long term connectivity results for one of the simulation runs are shown in Fig. 14. The same analysis is extended to the connectivity of the whole network W , with Z sub tree networks. Since node sets and links are disjoint in each tree, network survivability is the product of individual connectivity ratios,

$$p_C(W) = \prod_{k=1}^Z p_c(T_Z) \quad (4)$$

B. Monitoring Area Coverage

We have also analyzed the area coverage reliability $C(t)$ defined as the number of sensing areas covered, opposed to the number of nodes connected to the tree.

Under the assumption that sensing areas overlap with the transmission areas, losing some links may not necessarily mean losing the sensing area as shown in Fig. 12. At time t , tree T_1 has 8 nodes which are covering three sensing areas, whereas at time $t + \Delta t$ the same coverage ratio is preserved with only 5 nodes. The reverse may also be possible as losing a parent node may cause losing all subtrees of it. We have analyzed the long term coverage with five different independent availability rates for 10 consecutive queries. Fraction of areas covered versus long term reliability graph is illustrated in Fig. 13. Also, Fig. 15 compares the area coverage ratio with total connectivity for node availability rate of 0.99, which clearly shows a better performance so far as overlapping sensing ranges are concerned. We have extended the long term tree connectivity simulations for randomly generated events. Monitoring area consists of active and passive sensors where active sensors represent tree nodes having a valid path towards the root. Passive sensors represent dead links caused by battery run out. For each query time period we have randomly generated an event node in the monitoring area and tested the connectivity ratio of the event generating node. The connectivity results for these random events are also plotted in Fig. 15.

VI. CONCLUSION

It is inevitable that sensor networks will mature from small laboratory research testbeds to networks of millions of nodes, deployed densely to the field of interest. Such a dense environmental monitoring network in large scale geographic area will be the true enabler for the computational power that is taken granted anywhere

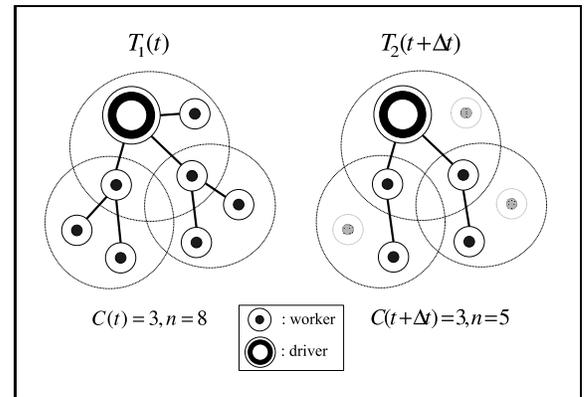


Fig. 12
MONITORING AREA COVERAGE

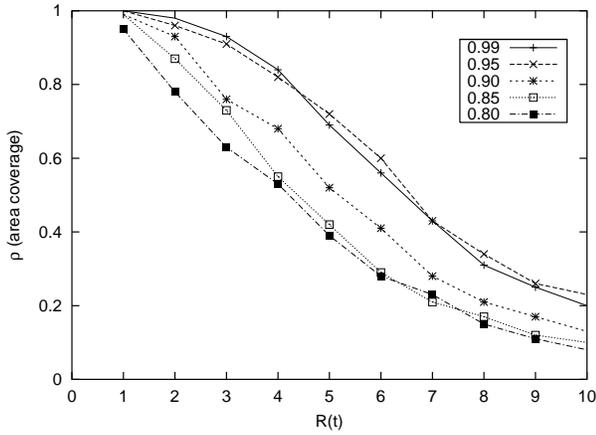


Fig. 13

NETWORK SURVIVABILITY (IN TERMS OF AREA COVERAGE)

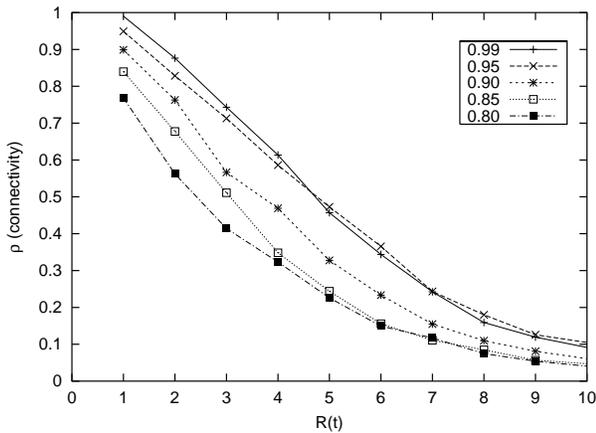


Fig. 14

NETWORK SURVIVABILITY (IN TERMS OF CONNECTIVITY)

[26]. This poses significant technical challenges. General purpose monitoring networks require adaptive cross layer protocol stacks for sensing devices as surveyed in [27]. Dynamic characteristics of sensor networks require radical changes in the design of current network protocols and monitoring applications. Network survivability is a very important criterion for deciding the efficiency of such protocols. This includes a measure of the network lifetime as well as the kind of service it provides during its lifetime. In this study, we have proposed a novel adaptive ad hoc network infrastructure with support for attribute based, named service access and dynamic sensor topology construction. The sensor network is

driven by the externally defined service parameters. The protocol stack for the service-aware backbone network, SeMA, was presented in [1]. Here, we have presented a monitoring service model with reference to XML based SeMA services and its use in the construction of a query to be run on the sensor network. The service information is disseminated from monitoring application to mobile client application with the intention to trigger responses from available sensors in the region. Interest query is distributed during topology construction. The proposed query driven data-centric routing algorithm's performance in maintaining a route to the sink is tested for tree connectivity/area coverage ratio and long term reliability. The proposed infrastructure provides a network solution for general purpose environmental monitoring applications with ad hoc and mobility requirements. Query driven sensor network behavior analysis were carried out via monte carlo simulations, demonstrating the long term availability of data return paths in the constructed topology. Simulations were based on independent node fail probabilities and random behavior of events. The MAC layer was abstracted away by assuming reliable links in each transmission range. The effect of contention for the medium during broadcast is ignored as it does not violate the internal working of the algorithms. End-to-end transmission is also supported by a dynamic path switching algorithm. This algorithm may be used to improve reliability between the event generating network and the driver node. Simultaneously, SeMA protocol stack simulation has been implemented into the Glo-MoSim network simulator [28]. The current work is the integration of the sensor nodes into the simulation to see the combined network performance in realizing a complete sensor network monitoring application. This way, sensor node simulations will not need to be skewed with MAC layer assumptions. Scalability and overhead characteristics of sensor networks will be analyzed with more accurate results. Throughput performance of the architecture for different event models such as tracking a moving target in the field or tracking frequent events generating a bursty data traffic could be investigated. Initial tests presented here are very encouraging and we believe that such a complete network solution may help the development of general purpose sensor network applications.

VII. ACKNOWLEDGMENT

The authors gratefully acknowledge student members of *The Network Laboratory*, for their help in drawing the figures and analyzing some simulation results.

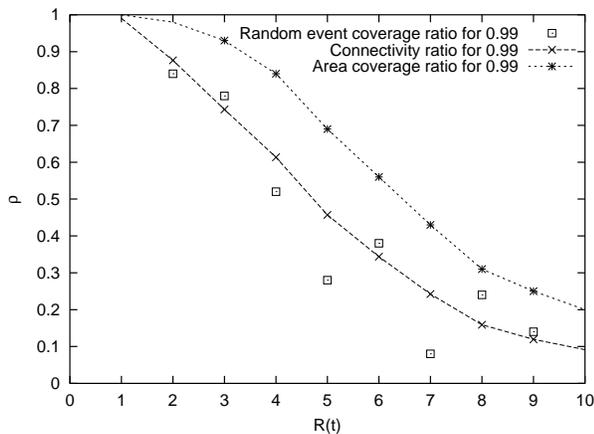


Fig. 15

COMPARISON OF CONNECTIVITY WITH AREA COVERAGE

REFERENCES

- [1] S. Baydere and M.A. Ergin, "An Architecture for Service Access in Mobile Ad Hoc Networks", *Proc. IASTED Wireless and Optical Communications Conference*, pp. 392–397, Banff, Canada, July 2002.
- [2] I. Rekleitis, G. Dudeck, and E. Milios, "Multi-robot Exploration of an Unknown Environment, Efficiently Reducing the Odometry Error", *Proc. Int. Joint Conf. on Artificial Intelligence*, pp. 1340–1345, Nagoya, Japan, August 1997.
- [3] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless Sensor Networks: A Survey", *Elsevier Computer Networks*, vol. 38, no. 4, pp. 393–422, March 2002.
- [4] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy Efficient Communication Protocols for Wireless Micro-Sensor Networks", *Proc. Hawaiian Int. Conf. on Systems Science*, vol. 8, January 2000.
- [5] R.C. Shah and J.M. Rabaey, "Energy Aware Routing for Low Energy Ad Hoc Sensor Networks", *Proc. IEEE WCNC*, pp. 350–355, Orlando, USA, March 2002.
- [6] P. Johansson, T. Larsson, N. Hedman, B. Mielczarek, and M. Degermark, "Scenario Based Performance Analysis of Routing Protocols for Mobile Ad Hoc Networks", *Proc. ACM/IEEE MOBICOM*, pp. 195–206, Seattle, USA, August 1999.
- [7] E.M. Royer and C.K. Toh, "A Review of Current Routing Protocols for Ad Hoc Mobile Wireless Networks", *IEEE Personal Communications*, vol. 6, no. 2, pp. 46–55, April 1999.
- [8] M. Lin, K. Marzullo, and S. Masini, "Gossip Versus Deterministic Flooding: Low Message Overhead and High Reliability for Broadcasting on Small Networks", *Tech. Rep. TR CS99-0637 UCSD*, November 1999.
- [9] M. Dorigo, V. Maniezzo, and A. Colomi, "The Ant System: Optimization by a Colony of Cooperating Agents", *IEEE Transactions on Systems, Man, and Cybernetics - Part B*, vol. 26, no. 1, pp. 1–13, 1996.
- [10] Y. Xu, J. Heidemann, and D. Estrin, "Geography-Informed Energy Conservation for Ad-hoc Routing", *Proc. ACM/IEEE MOBICOM*, pp. 16–21, Rome, Italy, July 2001.
- [11] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks", *Proc. IEEE/ACM MOBICOM*, pp. 56–67, Boston, USA, August 2000.
- [12] J. Kulik, W.R. Heinzelman, and H. Blakrishnan, "Negotiation-based Protocols for Disseminating Information in Wireless Sensor Networks", *Wireless Networks*, vol. 8, no. 2-3, pp. 169–185, 2002.
- [13] J. Heidemann, F. Silva, and C. Intanagonwiwat, "Building Efficient Wireless Sensor Networks with Low-level Naming", *Proc. Symposium on Operating Systems Principles*, pp. 146–159, Alberta, Canada, October 2001.
- [14] C.E. Perkins and P. Bhagwat, "Highly Dynamic Destination Sequenced Distance Vector Routing (DSDV) for Mobile Computers", *Proc. ACM SIGCOMM*, pp. 234–244, London, UK, October 1994.
- [15] T. Clausen and P. Jacquet, "Optimized Link State Routing", Internet Draft, <http://www.ietf.org/internet-drafts/drafts-ietf-manet-olsr-10.txt>, Work in progress, May 2003.
- [16] A. Cerpa et al., "Habitat Monitoring: Application Driver for Wireless Communications Technology", *Proc. ACM SIGCOMM*, pp. 20–41, Costa Rica, April 2001.
- [17] S. Sing, M. Woo, and C.S. Raghavendra, "Power Aware Routing in Mobile Ad Hoc Networks", *Proc. IEEE/ACM MOBICOM*, pp. 181–190, Dallas, USA, October 1998.
- [18] R. Jain, A. Puri, and R. Sengupta, "Geographical Routing for Wireless Ad Hoc Networks Using Partial Information", *IEEE Personal Communications*, vol. 8 no. 1, pp. 48–57, February 2001.
- [19] J. Chang and L. Tassiulas, "Energy Conserving Routing in Wireless Ad Hoc Networks", *Proc. IEEE INFOCOM*, pp. 22–31, Tel Aviv, Israel, March 2000.
- [20] S.R. Madden, M.J. Franklin, J.M. Hellerstein, and W. Hong, "TAG: A Tiny Aggregation Service for Ad Hoc Sensor Networks", *Proc. Symposium on Operating Systems Design and Implementation*, pp. 131–146, Boston, USA, December 2002.
- [21] J. Hill et al., "System Architecture Directions for Networked Sensors", *Proc. Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 93–104, Cambridge, USA, November 2000.
- [22] N. Sadagopan, B. Krishnamachari, and A. Helmy, "The ACQUIRE Mechanism for Efficient Querying in Sensor Networks", *Proc. Int. Workshop on Sensor Network Protocols and Applications*, pp. 149–155, Anchorage, USA, May 2003.
- [23] T. Bray, J. Paoli, C.M. Sperberg-McQueen, and E. Maler, "Extensible Markup Language (XML) 1.0 (Second Edition)", *W3C Recommendation*, October 2000.
- [24] S. Boag et al., "XQuery 1.0: An XML Query Language", *W3C Working Draft*, Work in progress, May 2003.
- [25] A. Kershenbaum, *Telecommunications Network Design Algorithms*, McGraw Hill Computer Science Series, 1993.
- [26] R. Min et al., "Energy-Centric Enabling Technologies for Wireless Sensor Networks", *IEEE Wireless Communications*, vol. 9, no. 4, pp. 28–39, August 2002.
- [27] A. Goldsmith and S.B. Wicker, "Design Challenges for Energy-Constrained Ad Hoc Wireless Networks", *IEEE Wireless Communications*, vol. 9, no. 4, pp. 8–27, August 2002.
- [28] L. Bajaj, M. Takai, R. Ahuja, R. Bagrodia, and M. Gerla, "GloMoSim: A Scalable Network Simulation Environment", *Tech. Rep. 990027 UCLA CSD*, May 1999.