A Self-Organizing Approach to Data Forwarding in Large-Scale Sensor Networks

Jelena Mirkovic, Geetha Priya Venkataramani,
Songwu Lu, Lixia Zhang
UCLA Computer Science Department
Los Angeles, CA 90095-1596
{sunshine, slu, lixia}@cs.ucla.edu

***Abstract* - The large number of networked sensors, frequent sensor failures and stringent energy constraints pose unique design challenges for data forwarding in wireless sensor networks. In this paper, we present a new approach to data forwarding in sensor networks that effectively addresses these design issues. Our approach organizes sensors into a dynamic, self-optimizing multicast tree-based forwarding hierarchy, which is data centric and robust to node failures. We demonstrate the effectiveness of our design through simulations.**

I. INTRODUCTION

Wireless sensor networks are envisioned as a new information technology that will provide target sensing, data collection, information manipulation and dissemination in a single integrated framework. Sensors will capture features or monitor activities of a set of objects (*stimuli*) in a sensor field, and report their observations to a set of interested clients (*sinks*) through the sensor network. Unlike traditional computer networks, sensor networks have frequently changing topology, all communications between nodes must be performed in a distributed manner, and individual nodes are very susceptible to sensor failures (due to energy depletion or destruction). The major challenge is the design of a scalable, self-adaptive sensor network that achieves high robustness in spite of low reliability of its components. Furthermore, data and connection replication cannot be used excessively in this case, since every additional message exchange consumes energy and increases the probability of sensor failures.

In this paper we present a new approach to data forwarding in a sensor network that effectively addresses some of these issues. Our protocol organizes sensors into a self-optimizing multicast tree-based forwarding infrastructure, which is data centric and robust to node failures. Our proposed approach is self-optimizing in the sense that it tends to minimize the data forwarding paths between multiple sources and sinks.

The rest of the paper is organized as follows. Section 2 describes the sensor network model and identifies key design challenges for data forwarding in such an environment. Section 3 presents the proposed multicast tree-based protocol and the node failure recovery algorithm. Section 4 provides simulation-based performance evaluations, Section 5 discusses related work, and Section 6 concludes this paper.

II. MODELS, ISSUES AND GOALS

A wireless sensor network is a wireless network that consists of a large number of sensors (say thousands) whose task is to monitor activities of a set of objects (*stimuli*) in a field, and report their observations to a set of interested clients (*sinks*). Sensors communicate with each other through the shared wireless channel. Note that a *sensor network* is different from the traditional *ad hoc network*, and communication principles used in one type of network are inapplicable to the other. The main difference is that the number of sensor components in a sensor network can be several orders of magnitude higher than the number of nodes in an ad hoc network. This makes scalability a critical design criterion and harder to address. Because of the large sensor population, no globally unique ID can be assigned to each sensor. Most of the data forwarding protocols devised for ad hoc networks either make use of a globally unique ID [8, 9] or assume small network size [7].

In the sensor network considered in this work, sensors are statically deployed in a possibly hostile, human-inaccessible environment, e.g., a battlefield; thus autonomous operation is a must. The computing and processing capability of each sensor is very limited, thus sophisticated and complex computations are deemed unrealistic. A sensor is able to communicate with its neighboring sensors over the shared wireless medium, but does not have global knowledge of the entire sensor network. The neighbors of a sensor are defined as all sensors within its transmission range, i.e., all sensors that can hear its data transmissions. Sensors are assumed to remain static throughout their lifetime. However, they may be *highly unreliable*, and they may fail without advance notice at any time. Sensors may also resume operations after failures in certain scenarios, e.g., when their solar batteries get recharged. Effectively, since sensors may "appear" or "disappear" dynamically, the whole network can be viewed as a dynamic graph with time-varying topology and connectivity. Since wireless transmissions are locally broadcast, the initiative is on the side of the receiving node, to accept or to reject a message. A decision is made based on the type of the message and the states of the receiving sensor.

Data forwarding protocol designed for such environments must address the following design issues:

- *Scalability*. Since the number of sensors in the field can be infinitely large, any global knowledge of the network is infeasible, and any global synchronization is unrealistic. Protocol for data forwarding must scale not only with the number of sensors but also with the number of sinks and stimuli.

- *Reliable delivery.* Reliable data delivery to the sinks must be ensured in spite of unreliable individual sensors.

- *Limited resources.* Sensors have limited energy supply, memory size and processing power. Their activity in data forwarding should thus be minimized in order to prolong their lifetime.

- *Error-prone wireless medium.* Sensors communicate through the wireless medium that is more error prone than the wired medium and has significantly lower bandwidth.

- *No globally unique ID.* Due to the large number of participating nodes, each node cannot be assigned a globally unique ID. Since there is no globally unique ID for each sensor, the addressing scheme cannot rely on the existence of a unique receiver for the message, and the routing protocol cannot use the reverse path for the message, at least not in the traditional way.

Consider the sensor network model and the issues to be addressed. The goals for an efficient data forwarding protocol can be identified as: the design of a reliable report delivery mechanism, which scales with the number of sinks, stimuli and intermediate sensors, and is robust to frequent sensor failures.

III. THE MULTICAST TREE PROTOCOL

In a typical application scenario, there may exist multiple clients (sinks) that are interested in receiving reports about a specific stimulus. The strawman approach is to send data to each client separately using multiple unicast delivery; this consumes a lot of energy unnecessarily in the case when sinks are spatially co-located and share the same interest. The proposed optimization is to organize the sinks and the stimulus into a multicast tree and to use this infrastructure for the distribution of report messages. The minimum spanning tree would be the optimal structure that minimizes the data-path length for a static graph. The dynamically changing topology and lack of global topology information make this approach infeasible. Therefore, our proposed protocol seeks to construct the tree online and to optimize it through local mechanisms.

In the absence of sink interests, sensors lie in the field inactive, and periodically switch on to listen to the advertisement messages. This way, energy is not wasted unnecessarily. Once sinks develop an interest in a certain stimulus, they go through the advertisement process in which the multicast tree is constructed. In the absence of stimuli, this tree is maintained by exchanging low-frequency *hello* messages between sensors on the tree. Once a stimulus arrives in the field, a new branch is formed that connects this stimulus to the tree, and report generation and data forwarding are triggered. At this stage, reports are used as *hello* messages to maintain the tree structure. If after some time a sink loses interest in the stimulus, it tears down its branch. If all sinks tear their branches, the branch towards the stimulus is torn down, and all sensors along this branch become inactive again.

The protocol consists of four phases for building a multicast tree. The detailed steps are explained next and illustrated in Fig. 1.

A. *First phase: Broadcasting sink advertisements*

When a sink declares interest in certain type of stimulus, it initiates the construction of the multicast tree by broadcasting a *SinkAdvertisement* message stating its *SinkID*. Each message also carries *nofhops*, the number of hops that a message was forwarded before it reached the specific sensor. It is

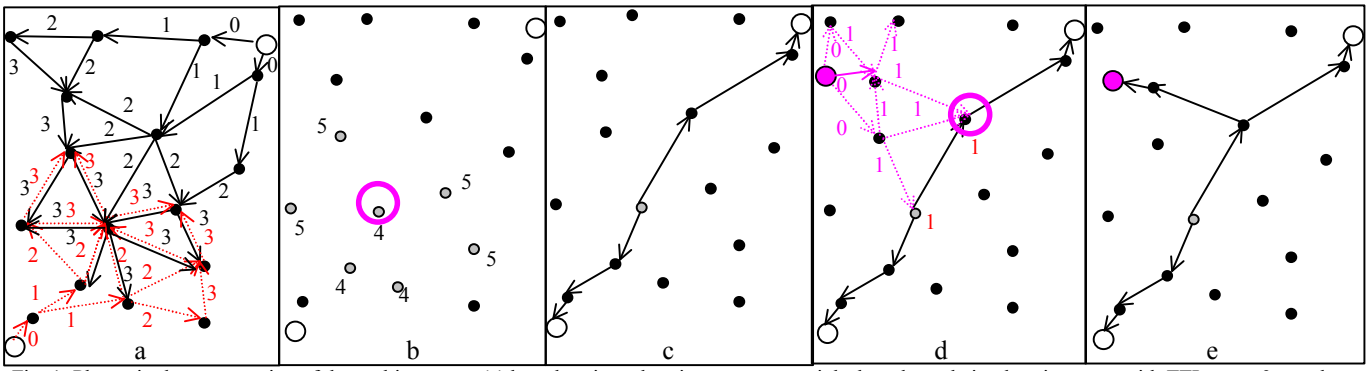

Fig. 1: Phases in the construction of the multicast tree: (a) broadcasting advertisements - two sinks broadcast their advertisements with TTL set to 3, numbers represent *nofhops* values in advertisements, (b) electing the merge point – light gray circles represent candidate merge points and numbers their cumulative distances, circled point is elected, (c) forming the tree, (d) broadcasting the new node advertisements with TTL=1, numbers next to the nodes represent the distance of tree nodes to the stimulus, circled point is elected to graft the new node, and (e) grafting the node.

initially set to 0 by the sink, and gets incremented by each sensor that forwards the message. Forwarding of this message may engage all the sensors in the field and consume the energy. Sink can avoid this by setting a *TTL* value for the message – the maximum number of hops this message can be forwarded, thus limiting the number of sensors that participate in message forwarding.

B. Second phase: Election of candidate merge points

A sensor that receives the *SinkAdvertisement* message, stores *SinkID* and *nofhops* from this message in its Sink Table. The sensor then forwards the *SinkAdvertisement* further if *nofhops* in the message is less than *TTL* value. Otherwise, the message is dropped. The sensor also starts a timer upon whose expiration it goes into the third phase.

C. Third phase: Election of one merge point

When the timer expires, if the sensor is a candidate merge point for more than one sink in its Sink Table, it initiates the negotiation phase by broadcasting a *MergeAdvertisement* message in which it states the sum of distances from itself to all sinks it received *SinkAdvertisement* message from, and a set of their *SinkIDs*. A timer is set upon transmission of the message in order to allow all candidate merge points to send their advertisements.

Only sensors that are also candidate merge points listen to *MergeAdvertisement* messages. If the set of sinks that they have heard from is smaller than the advertised set, or their cumulative distance is larger, they forward the message to their neighbors. Otherwise, the message is silently dropped.

D. Fourth phase: Branch formation towards the sinks

When the timer expires, the sensor that did not hear any advertisement better than its own, considers itself a Merge Point and sends *MergeAcknowledgment* towards all sinks in its sink table. On the way towards the sinks, the *MergeAcknowledgment* message builds the branches of the tree by setting a state in each sensor that forwards the message. This state consists of IDs of the upstream and downstream neighbors. Even though this implies the need for having a sensor ID, this ID need not be globally unique. It is sufficient that all neighbors of a sensor have different IDs; two sensors may still have the same IDs if they do not have common neighbors.

E. Maintaining the tree: Keepalive mechanism

Once the tree is formed, nodes on the tree send keep-alive messages to their neighbors at very low frequency. Each stimulus report resets the *KeepAlive* timer to 0 (so reports are used as keepalive messages).

F. Forming a new branch: Joining of the new node

Once a new stimulus appears in the field or a new sink declares interest in an existing stimulus, there is a need for connecting this new node to the tree. To this end, we elect the sensor on the tree that is closest (measured in the number of hops) to the new node and form a branch from this sensor to the new node.

A sink or a stimulus that intends to join the tree goes through the advertisement process, the same as the one described in the first phase. Advertisement messages are forwarded until they reach the tree. After that, a negotiation is performed between tree nodes to find the one closest to the new node. This tree node then sends a *MergeAcknowledgment* that creates the branch towards the new node.

G. Report forwarding

Once the tree is formed and each of the branches leads to a stimulus, the process of report generation and forwarding begins. The stimulus periodically generates new reports and sends them along the tree. Each node on the tree receives the report from its upstream neighbor and forwards it to its other neighbors on the tree. Thus both the number of sensors that take part in report forwarding and the communication overhead are minimized. Moreover, each routing table consists only of upstream and downstream IDs. Its size is dependent on the density of sensors in the field, but not on the number of sinks or stimuli. This definitely scales well in the presence of a large number of sinks.

H. Tree damage and reconnection

The weak point of the tree structure is its sensitivity to failures. This demands an efficient node recovery mechanism, which can handle node failures that affect a single sensor or a group of sensors. Node failures can often occur in the following two scenarios: (1) catastrophic event that destroys the node, or (2) energy depletion.

In the case of sudden node failure detection is performed by using timers and passive acknowledgment. Every node monitors the messages it has sent to its neighbors and checks if they are forwarded. If a neighbor did not forward any message within some time interval, it is considered dead and the sensor attempts to reconnect the tree. The tree is reconnected through the advertisement process, similar to the one described for tree formation. TTL in the advertisements is set to several hops and disconnected end points of the tree initiate reconstruction and act as sinks.

In the case of node failures due to energy depletion, a smooth hand-off from a low-energy sensor that is a part of the multicast tree can be performed. The failing sensor informs its upstream and downstream neighbors of its failure,

and they try to reconnect the tree by choosing the best of their common neighbors. If there are no common neighbors, the tree is reconnected in the same manner as in the first case.

IV. PERFORMANCE EVALUATION

We evaluate our design through simulations. The simulator is written in Parsec, a C language with message-passing mechanisms. The number of sensor reports generated (per report delivered to the sink) is used as the performance measure. Since every message consumes sensor energy, a smaller number of reports means less power consumed in the test time interval.

We compare four forwarding algorithms:

1. *Reverse path forwarding (RPF).* The number of hops is used for addressing. A sensor sends a message with the number of hops N, and all sensors in its neighborhood that have distance N hops from the desired sink will pick up the message and broadcast it with number of hops N-1. This is a broadcast scheme.

2. *Truncated reverse path forwarding (TRPF).* All sensors in the neighborhood are dynamically assigned locally unique addresses. Each sensor stores the address of its next hop to sink (previous hop that forwarded to it the sink advertisement message) and sends the message directly to this sensor. This is a unicast scheme and it requires a lot of storage for the large number of sinks.

3. *Merge point initiated multicast (MPIM).* The algorithm that is proposed in this paper.

4. *Sink initiated multicast (SIM).* The same algorithm as merge point initiated multicast, except that the merge point is co-located with one of the sinks. The purpose of simulating this algorithm is to demonstrate the impact of the position of the merge point on the algorithm's efficiency.

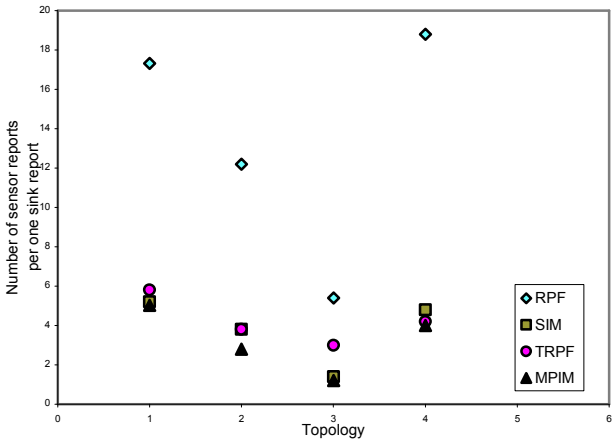

Fig. 2: Simulation results for four different topologies

We simulate different topologies to evaluate the efficiency of each algorithm. The results are shown in Fig. 2.

From simulations, we can draw the following conclusions:

1. The reverse-path forwarding scheme always has large overhead, in some cases even 10 times larger than other approaches.

2. The truncated reverse path based scheme has performance comparable to the multicast-based approach, but it has large storage requirements.

3. The sink-initiated multicast scheme generates up to 30% more reports per one sink report, compared to the merge point initiated multicast scheme. This might not be critical for short-term transmissions. However, for long-term transmissions over a large time interval, this would mean that sensors only have 76% of their lifetime when the tree is optimized. Thus optimization definitely pays off for long-term transmissions.

V. RELATED WORK

In recent years, a lot of algorithms have been designed for on-demand dynamic routing in ad hoc networks [7, 8, 9]. However, approaches in [8] and [9] cannot be applied to the sensor network since they assume the existence of a unique ID for each node. The approach suggested in [7] is designed for small-size networks and would not scale well if applied to sensor networks. In [4], a directed diffusion paradigm is proposed for report forwarding in sensor networks. Nodes organize themselves into clusters with the cluster head being dynamically elected to perform report forwarding. Periodically, the cluster head is reelected to optimize the energy consumption among nodes in the cluster. This approach promises robustness but has a drawback when the synchronization is lost during the election process, namely competing nodes may enter deadlock if they all promote themselves to the next level simultaneously. In [6], the directed diffusion algorithm is refined and implemented in a simulation environment. Simulations have shown energy efficiency of this design over the multicast. However, the only simulated scenario was the one in which source reports were not aggregated during multicast, but aggregation was performed during directed diffusion. We have not been able to compare the algorithm in [6] with our design at this moment. In [5], the problem of energy-efficient data delivery is studied but the paper does not consider the issues of scalability and node failures explicitly.

VI. CONCLUSIONS

Emerging wireless sensor networks are featured with a significantly large number of sensors, and individual sensor failures may become a norm rather than an exception. The design of a sensor networking system for efficient data forward-

ing poses severe design challenges. For such a system to be feasible and effective, it must provide exception-free, unattended operations. The designed protocol should be self-configuring and robust to frequent changes in the network condition. In this paper, we have described a multicast-tree based protocol for data forwarding in a sensor network, and we also proposed mechanisms to effectively handle node failures in this protocol. Our simulation results show the effectiveness of our approach. Ongoing and future work seeks to refine the multicast tree-based algorithm to handle sensor mobility, perform more extensive simulations and analytically characterize the performance of the proposed design.

REFERENCES

[1] G. Pottie, W. Kaiser, L. Clare, and H. Marcy. "Wireless integrated network sensors: Towards low cost and robust self-organizing security networks," *UCLA Technical Report*, 1999.

[2] R. Govindan, T. Faber, J. Heidemann, and D. Estrin, "Ad hoc smart environments", USC Technical Report 99-692, 1999.

[3] T.J. Shepard, "A channel access scheme for large dense packet radio networks," *ACM SIGCOMM'97*, 1997.

[4] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar, "Next century challenges: Scalable coordination in sensor networks," *ACM MOBICOM'99*, 1999.

[5] W. R. Heinzelman, J. Kulik, and H. Balakrishnan, "Adaptive protocols for information dissemination in wireless sensor networks," *ACM MOBICOM'99*, 1999.

[6] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed diffusion: A scalable and robust communication paradigm for sensor networks," *ACM MOBICOM'00*, 2000.

[7] D. Johnson and D. Maltz, "Dynamic source routing in ad hoc wireless networks," *Mobile Computing, pp.153-181*, 1996.

[8] V. Park and M. Corson, "A highly adaptive distributed routing algorithm for mobile wireless networks" in *Proceedings of INFOCOM'97*, 1997.

[9] C. Perkings, "Ad hoc on demand distance vector routing", *Internet Draft,* July 2000.