ASCENT: Adaptive Self-Configuring sEnsor Networks Topologies.

Alberto Cerpa and Deborah Estrin.
Department of Computer Science, University of California, Los Angeles
3440 Boelter Hall, Los Angeles, CA 90095
{cerpa, destrin}@cs.ucla.edu

Abstract— Advances in micro-sensor and radio technology will enable small but smart sensors to be deployed for a wide range of environmental monitoring applications. The low per-node cost will allow these wireless networks of sensors and actuators to be densely distributed. The nodes in these dense networks will coordinate to perform the distributed sensing tasks. Moreover, as described in this paper, the nodes can also coordinate to exploit the redundancy provided by high density, so as to extend overall system lifetime. The large number of nodes deployed in these systems will preclude manual configuration, and the environmental dynamics will preclude design-time pre-configuration. Therefore, nodes will have to self-configure to establish a topology that provides communication and sensing coverage under stringent energy constraints. In ASCENT, each node assesses its connectivity and adapts its participation in the multi-hop network topology based on the measured operating region. This paper motivates and describes the ASCENT algorithm and presents simulation and experimental measurements.

I. INTRODUCTION

The availability of micro-sensors and low-power wireless communications will enable the deployment of densely distributed sensor/actuator networks for a wide range of environmental monitoring applications from urban to wilderness environments; indoors and outdoors; and encompassing a variety of data types including acoustic, image, and various chemical and physical properties [26]. The sensor nodes will perform significant signal processing, computation, and network self-configuration to achieve scalable, robust and long-lived networks [1, 7, 8]. More specifically, sensor nodes will do local processing to reduce communications, and consequently, energy costs.

These requirements pose interesting challenges for networking research. One of the challenges arises from the greatly increased level of *dynamics***.** The large number of nodes will introduce increased levels of *system dynamics*, which in combination with the high level of *environmental dynamics* will make designing reliable systems a daunting task. Perhaps the most important technical challenge arises from the *energy constraints* imposed by *unattended systems*. These systems must be long-lived and operate without manual intervention, which implies that the system itself must execute the measurement and adaptive configuration in an energy constrained fashion. Finally, there are *scaling* challenges associated with the large numbers of nodes that will co-exist in such networks to achieve desired spatial coverage and robustness.

In this paper, we describe and present simulation and experimental performance studies for a form of adaptive self-configuration designed for sensor networks. As we argue in Section II, such unattended systems will need to self-configure and adapt to a wide variety of environmental dynamics and terrain conditions. These conditions produce regions with non-uniform communication density. We suggest that one of the ways system designers can address such challenging operating conditions is by deploying redundant nodes and designing the system algorithms to make use of that redundancy over time to extend the systems life. In ASCENT, each node assesses its connectivity and adapts its participation in the multi-hop network topology based on the measured operating region. For instance, a node:

- Signals when it detects high message loss, requesting additional nodes in the region to join the network in order to relay messages.
- Reduces its duty cycle if it detects high message losses due to collisions.
- Probes the local communication environment and does not join the multi-hop routing infrastructure until it is ”helpful” to do so.

Why can this adaptive configuration not be done from a central node? In addition to the scaling and robustness limitations of centralized solutions, a single node cannot directly sense the conditions of nodes distributed elsewhere in space. Consequently, other nodes would need to communicate detailed information about the state of their connectivity in order for the central node to determine who should join the multi-hop network. In the absence of energy constraints, one can always achieve a result that is closer to optimal with a central computation. However, when energy is a constraint and the environment is dynamic, distributed approaches are attractive and possibly are the only practical approach [21] because they avoid transmitting dynamic state information repeatedly across the network.

Pottie and Kaiser [21] initiated work in the general area of wireless sensor networks by establishing that scalable wireless sensor networks require multi-hop operation to avoid sending large amounts of data over long distances. They went on to define techniques by which wireless nodes discover their neighbors and acquire synchronism. Given this basic bootstrapping capability, our work addresses the next level of automatic configuration that will be needed to realize envisioned sensor networks, namely, how to form the multi-hop topology [8]. Given the ability to send and receive packets, and the objective of forming an energy-efficient multi-hop network, we apply well-known techniques from MAC layer protocols to the problem of distributed topology formation. Similar techniques have been applied to multicast transport protocol adjustment of periodic messaging [9, 10].

In the following section we present a sensor network scenario, stating our assumptions and contributions. Section III describes ASCENT in more detail. In Section IV, we present some initial simulation and experimental results using ASCENT. Related work is reviewed in Section V.

II. Distributed Sensor Network Scenario

To motivate our research, consider a habitat monitoring sensor network that is to be deployed in a remote forest. Deployment of this network can be done, for example, by dropping a large number of sensor nodes from a plane, or placing them by hand. In this example, and in many other anticipated applications of ad-hoc wireless sensor networks, the deployed systems must be designed to operate under the following conditions and constraints:

- **Ad-hoc deployment**: we cannot expect the sensor field to be deployed in a regular fashion (e.g. a linear array, 2-dimensional lattice). More importantly, uniform deployment does not correspond to uniform connectivity owing to unpredictable propagation effects when nodes, and therefore antennae, are close to the ground and other surfaces.
- **Energy constraints**: The nodes (or at least some significant subset) will be untethered for power as well as communications and therefore the system must be designed to expend as little energy as is possible in order to maximize network lifetime.
- **Unattended operation under dynamics**: the anticipated number of elements in these systems will preclude manual configuration, and the environmental dynamics will preclude design-time pre-configuration.

In many such contexts it will be far easier to deploy larger numbers of nodes initially than to deploy additional nodes or additional energy reserves at a later date (similar to the economics of stringing cable for wired networks). In this paper we present one way in which nodes can exploit the resulting redundancy in order to extend system lifetime.

If we use too few of the deployed nodes, the distance between neighboring nodes will be too great and the packet loss rate will increase; or the energy required to transmit the data over the longer distances will be prohibitive. If we use all deployed nodes simultaneously, the system will be expending unnecessary energy, at best, and at worst the nodes may interfere with one another by congesting the channel. In the process of finding an equilibrium, we are not trying to use a distributed localized algorithm to identify a single optimal solution. Rather this form of adaptive self-configuration using localized algorithms is well suited to problem spaces that have a large number of possible solutions; in this context a large solution space translates into dense node deployment. Our simulation and experimental results confirm that this is the case for our application.

We enumerate the following assumptions that apply to the remainder of our work:

- We assume a Carrier Sense Multiple Access (CSMA) MAC protocol with capacity to work in promiscuous mode. This clearly introduces the possibilities for resource contention when too many neighboring nodes participate in the multihop network. Our approach should be relevant to TDMA MACs as well because distributed slot allocation schemes will also have degraded performance with increased load. Future work will investigate the use of ASCENT with other MAC protocols under development [31].
- Our algorithm reacts when links experience high packet loss. The ASCENT mechanism does not detect or repair network partitions. Partitions are more prevalent when node density is low, and our approach is not applicable because in general all nodes will be needed to form an effective network. Of course network partitions can occur even in dense arrays when a swath of nodes are destroyed or obstructed. When such network partitions do occur, complementary system mechanisms will be needed; for example, detecting partitions in the multi-hop sensor network by exploiting information from long range radios deployed on a subset of nodes, and used sparingly because of the power required. We leave such complementary techniques for network partition detection and repair to future work.

The two primary contributions of our design are:

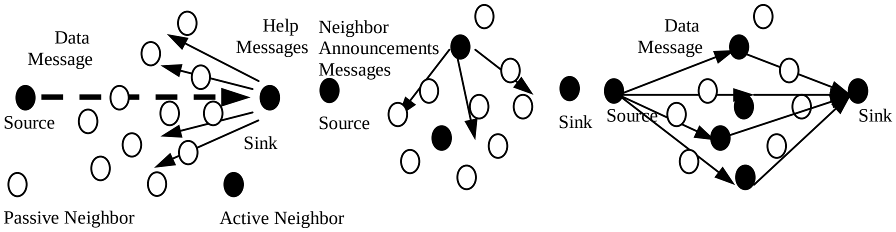
- The use of adaptive techniques that permit applications to ***configure*** the underlying topology based on their needs while trying to save energy to extend network lifetime. Our work does not presume a particular model of fairness, degree of connectivity, or capacity required.
- The use of self-configuring techniques that react to operating conditions ***measured locally***. Our work is not restricted to the radio propagation model, the geographical distribution of nodes, or the routing mechanisms used.

The following section describes the ASCENT protocol in some detail.

III. ASCENT Design

ASCENT adaptively elects "active" nodes from all nodes in the network. Active nodes stay awake all the time and perform multi-hop packet routing, while the rest of the nodes remain "passive" and periodically check if they should become active.

Consider a simple sensor network for data gathering similar to the network described in Section 2. We cannot expect the sensor field to have uniform connectivity due to unpredictable propagation effects in the environment. Therefore, we would expect to find regions with low and high density. As we pointed out in Section 2, ASCENT does not deal with complete network partitions; we assume that there is a high enough node density to connect the entire region. Fig. 1 shows a simplified schematic for ASCENT during initialization in a high-density region. For the sake of clarity, we show only the formation of a two-hop network. This analysis may be extended to networks of larger sizes.



(a) Communication Hole (b) Transition (c) Final State

Fig. 1. Self-configurable sensor network.

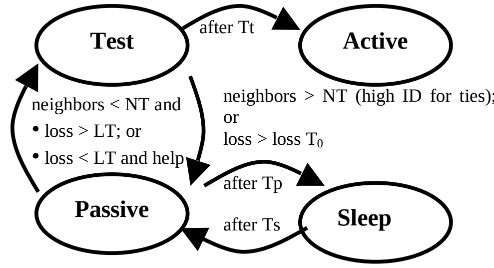


Fig. 2. State transitions in ASCENT

Initially, only some nodes are active. The other nodes remain passively listening to packets but not transmitting. This situation is depicted in Fig. 1(a). The source starts transmitting data packets toward the sink. Because the sink is at the limit of radio range, it gets very high message loss from the source. We call this situation a *communication hole;* the receiver gets high packet loss due to poor connectivity with the sender. The sink then starts sending *help messages* to signal neighbors that are in listen-only mode –also called *passive neighbors*– to join the network.

When a neighbor receives a *help message*, it may decide to join the network. This situation is illustrated in Fig. 1(b). When a node joins the network it starts transmitting and receiving packets, i.e. it becomes an *active neighbor*. As soon as a node decides to join the network, it signals the existence of a new active neighbor to other passive neighbors by sending a *neighbor announcement message*. This situation continues until the number of active nodes stabilizes on a certain value and the cycle stops. When the process completes, the group of newly active neighbors that have joined the network make the delivery of data from source to sink more reliable. The process will re-start when some future network event (e.g. node failure) or environmental effect (e.g. new obstacle) causes message loss again.

In this section, we describe the ASCENT algorithm and their components. Several design choices present themselves in this context. We elaborate on these design choices while we describe the design. Our initial simulations, and experiments (Section 4) focus only on a subset of these design choices.

A. ASCENT state transitions

In ASCENT, nodes are in one of four states: *sleep, passive, test, and active.* Fig. 2 shows a state transition diagram.

Initially, a random timer turns on the nodes to avoid synchronization. When a node starts, it initializes in the *test state*. Nodes in the *test state* exchange data and routing control messages. In addition, when a node enters the *test state*, it sets up a timer Tt, and sends neighbor *announcement* messages. When Tt expires, the node enters the *active state*. If before Tt expires the number of active neighbors is above the *neighbor threshold* (NT), or if the average *data loss rate* (DL) is higher than the average loss before entering in the *test state*, then the node moves into the *passive state*. If multiple nodes make a transition to the test state, then we use the node ID in the announcement message as a tie breaking mechanism (higher IDs win). The number of active nodes cannot exceed the NT value. The intuition behind the *test state* is to probe the network to see if the addition of a new node may improve connectivity.

When a node enters the *passive state*, it sets up a timer Tp. When Tp expires, the node enters the *sleep state*. If before Tp expires the number of neighbors is below NT, and either the DL is higher than the *loss threshold* (LT) or DL is below the *loss threshold* but the node received a *help message* from an active neighbor, it makes a transition to the *test state*. While in *passive state* nodes have their radio on, and are able to overhear all packets transmitted by their active neighbors (even if the packets are not addressed to the passive node, the radio is in promiscuous mode). No routing or data packets are forwarded in this state, since this is a listen-only state. The intuition behind the *passive state* is to gather information regarding the state of the network without causing interference with the other nodes. Nodes in the *passive* and *test states* continuously update the number of active neighbors and *data loss rate* values. Energy is still consumed in the *passive state*, since the radio is still on when not receiving packets. A node that enters the *sleep state* turns the radio off, sets a timer Ts and goes to sleep. When Ts expires, the node moves into *passive state*. Finally, a node in *active state* continues forwarding data and routing packets until it runs out of energy. If *data loss rate* is greater than LT, the active node sends *help messages*.

B. ASCENT parameters tuning

ASCENT has many parameters that can affect its final behavior. Parameters like the timers Tt, Tp, Ts, *neighbor threshold* (NT), and *loss threshold* (LT) are choices left to the applications. In this section, we explain the choices made in the current ASCENT algorithm. A particular application may select different parameter settings, for instance, perhaps trading energy savings for greater sensing coverage.

The *neighbor threshold* (NT) value determines the average degree of connectivity of the network. When the sensing range of the micro-sensors is lower than the radio communication range, the minimum number of nodes necessary to provide communication coverage may not be sufficient for sensing coverage. There is a trade-off between the energy consumed and/or the level of interference (packet loss) vs. the desired sensing coverage. An application could adjust this value dynamically depending on the events occurring in a certain area of the network. In this study, we set this value to 4.

The *loss threshold* (LT) determines the maximum amount of data loss an application can tolerate before it requests help to improve network connectivity. This value is very application dependent. For example, average temperature measurements from a sector of a forest will not tend to vary drastically, and the application may tolerate high packet loss. In contrast, tracking of a moving target by the sensor network may be more sensitive to packet losses. In our implementation this value was set to 20%.

The test timer Tt and the passive timer Tp determine the maximum time a node remains in the test and passive states, respectively. They face a similar trade-off of power consumption vs. decision quality. The larger the timers, the more robust the decision in presence of transient packet losses (that also affects the neighbor determination, see next section), but the greater the power consumed with the radio on, and vice versa. Similarly, the sleep timer Ts represents the amount of time the node sleeps to preserve energy. The larger the Ts timer, the larger the energy savings, but the larger also the probability of no node listening when necessary (slower reaction time to dynamics). In our implementation, the relation between these timers was set to 2:5:30 for Tt, Tp, and Ts respectively. The final choice of timer values should be dependent on the application needs.

In the sleep state, nodes turn their radios off and sleep for a certain amount of time. This is only one possible form of adaptation. For example, nodes could also reduce their radio range, using less energy and reducing channel contention. The experimental platform we used for our experiments did not have this capability, but we plan to repeat the experiments using more capable hardware.

C. Neighbor and Data Loss determination

The number of active neighbors and the average data loss rate are values measured *locally* by each node while in *passive* and *test state*.

The number of neighbors can greatly increase the energy consumption in contention for resources. We have chosen to define a neighbor as a node from which we receive a certain percentage of packets over time. This implies having a history function that keeps track of the packets received from each individual node over a certain period (time and/or number of messages), and a fixed or dynamic *neighbor loss threshold*.

In ASCENT, each node adds a unitary monotonically increasing sequence number to each packet transmitted (including data and control packets). This permits neighbor link loss detection when a sequence number is skipped. In addition, we assume application data packets also have some mechanism to detect losses (data payload sequence numbers in our implementation).

The number of active neighbors N is defined as the number of neighbors with link packet loss smaller than the neighbor loss threshold.

We have chosen the following formula for *neighbor loss threshold* (NLS):

$$NLS = 1 - \frac{1}{N} \qquad (1)$$

with N being the number of neighbors calculated in the previous cycle.

When a node perceives a neighbor's packet loss (i.e. the link packet loss calculated for all the packets sent by that neighbor) larger than the NLS, it no longer considers that node as a neighbor and deletes it from its neighbor list. The intuition behind this formula is the following: as we increase the number of neighbors in the region, the likelihood of any pair of them not listening to each other (or having high losses) increases. Therefore, as we increase the number of neighbors, we should correspondingly increase the neighbor's loss threshold. Not doing so may result in getting a lower neighbor count even though nodes in the region may still interfere with each other. Correspondingly, as we decrease the number of neighbors, we should decrease the neighbor's message loss threshold accordingly. (We experimented with some other functions, like an inversely decaying function of 1/N and an exponentially decaying function of 1/N; but the simple formula above worked best).

The average *data loss rate* (DL) is calculated based on the application data packets. Data losses are detected using data sequence numbers. Depending on the routing strategy, a node *may* receive multiple copies of the same application data packet. We only consider a data loss if the message was not received from any neighbor during a certain configurable period of time (this allows out of order delivery based on the application needs). Control messages (help, neighbor announcements and routing) are not considered in this calculation.

D. ASCENT interactions with routing

ASCENT runs above the link and MAC layer and below the routing layer. ASCENT is not a routing or data dissemination protocol. ASCENT simply decides which nodes should join the routing infrastructure. Ad-hoc routing [15, 18, 20], Directed Diffusion [13], or some other data dissemination mechanism, then runs over this multihop topology. In this respect, routing protocols are complementary to ASCENT.

ASCENT nodes become active or passive independent of the routing protocol running on the node. In addition, ASCENT does not use state gathered by the routing protocol, since this state may vary greatly for different protocols (e.g. ad-hoc routing tables & directed diffusion gradients), or requires changing the routing state in any way. Currently, if a node is testing the network and it is actively routing packets when it becomes passive, ASCENT depends on the routing protocol to quickly re-route traffic. This may cause some packet loss, and therefore an improvement that has not been implemented is to inform the routing protocol of ASCENT's state changes so traffic could be re-routed in advance.

We emphasize that, even though we have discussed the ASCENT algorithm in some detail, much experimentation and evaluation of the various mechanisms and design choices is necessary before we fully understand the robustness, scale and performance of self-configuration. The following section presents our initial findings based on simple analysis, simulation, and an experimental implementation.

IV. PERFORMANCE EVALUATION

In this section, we report results from a preliminary performance evaluation of ASCENT. We use a simple mathematical model to determine an idealized level of packet loss, delay, and energy savings as we increase node density. Since our analysis cannot capture the complexity of a full ASCENT scenario, we use simulations and real experiments to further validate the performance evaluation.

A. Analytic performance analysis

To understand the relation between packet collisions and density of nodes we first use a simple mathematical analysis.

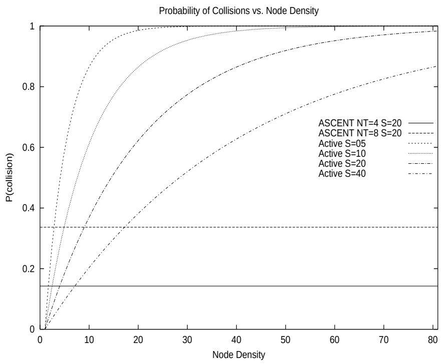


Fig. 3. P(collisions) vs. density

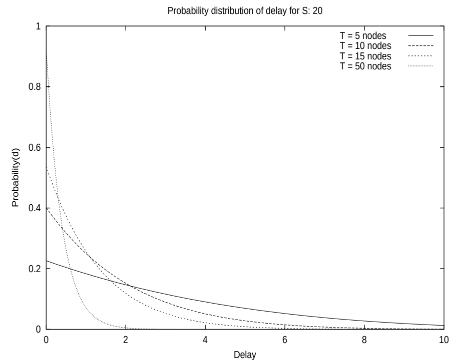


Fig. 4. Delay prob. distribution

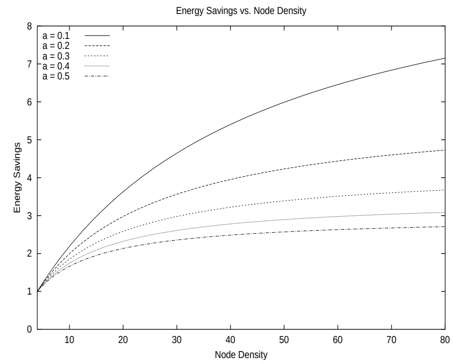


Fig. 5. Energy Savings vs. density

Assume that the communication range of each node is R, and the number of nodes per πR^2 area is n. Further assume a simple CSMA MAC with a random back-off. This random component is chosen from a discrete pool of S slots with a uniform probability distribution.

Thus, the probability of collisions for T nodes transmitting in the area of radius R is given by:

$$P(collision) = 1 - P(success) = 1 - \left(\frac{S-1}{S}\right)^{T-1} \qquad (2)$$

From this formula we see that as we increase the density of transmitting nodes T, the probability of collisions increases proportionally. When all the nodes in the network are able to transmit and receive packets, we find that T = n, since every node in the πR^2 area can transmit packets. Increasing the density of nodes increases the probability of collisions in the area. ASCENT fixes the number of transmitters in the area to the *neighbor threshold* (NT) value, resulting in T = NT, independent of the total number of nodes, n, deployed. Fig. 3 shows the analytical relation between packet losses vs. density of nodes for different S and NT values.

The relation between the hop-by-hop delay introduced by the randomization and the density of nodes can be analyzed similarly. The average delay experienced per hop is related to the number of random slots S and the total number of active nodes T. After reception of a message to be forwarded toward the destination, each of the T active nodes picks a random slot, say $S_1, S_2, \ldots, S_T$. The mean number of all the random slots chosen will tend to be S/2, since it is a uniform probability distribution. Assuming no collision losses, i.e.: $\forall i \neq j \in 1,2,\ldots,T \Rightarrow S_i \neq S_j$, the hop-by-hop delay is determined by the first message to be forwarded. The delay d is then:

$$d = \min(S_1, S_2, \ldots, S_T) \qquad (3)$$

We want to find P(d), the probability distribution of the smaller random time slot picked by T nodes. We define:

$$Q(y) = \mathrm{Prob}\left[\min(S_1, S_2, \ldots, S_T) > y\right] \qquad (4)$$

$$\min(S_1, S_2, \ldots, S_T) > y \Leftrightarrow \text{each of } (S_1, S_2, \ldots, S_T) > y$$

This happens with probability

$$\left(\frac{(S-y)}{S}\right)^T \;\therefore\; Q(y) = \left(1 - \frac{y}{S}\right)^T \qquad (5)$$

P(d) as we defined above is:

$$P(d) = Q(d) - Q(d+1) = \left(1 - \frac{d}{S}\right)^T - \left(1 - \frac{(d+1)}{S}\right)^T \qquad (6)$$

Fig. 4 shows the P(d) distribution for different values of T and S=20. When all the nodes in the network are able to transmit and receive packets, we find that T = n. As n increases, the mean value of P(d) decreases. This result corresponds to the intuition that as we increase the total number of transmitting nodes, the likelihood of any of them picking a smaller random value increases. In the ASCENT case, T = NT independently of the density n, and the mean value of P(d) remains constant. Finally, we would like to understand the energy savings that could be obtained by using ASCENT. When the system is not running ASCENT, all the nodes have their radios on, consuming *Idle* power[1]. When the system is running ASCENT, NT nodes have their radios on, while the rest alternate between sleeping and listening. The energy savings (ES) are:

$$\frac{n * Idle}{NT * Idle + (n - NT) * Idle * \dfrac{Tp}{Tp + Ta} + (n - NT) * Sleep * \dfrac{Ta}{Tp + Ta}}$$

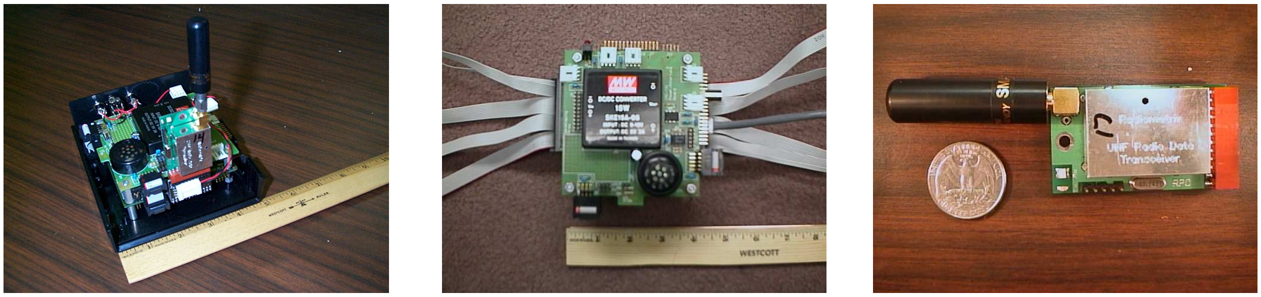
The 2^{nd} term in the denominator indicates the energy of non-active nodes when in passive state, and the 3^{rd} term indicates the energy consumed while in the sleep state. We define α to be the ratio of the passive timer Tp to the sleep timer Ts. We also define β to be the ratio of the power consumed by the radio in sleep mode to the power consumed by the radio in idle mode. Then, we obtain the following equation for ES:

$$ES = \frac{n}{NT + (n - NT) * \dfrac{\alpha + \beta}{\alpha + 1}} \qquad (7)$$

$$\lim_{n \to \infty} ES = \frac{\alpha + 1}{\alpha + \beta} \qquad (8)$$

Fig. 5 shows the energy savings as we increase the density of nodes for a fixed value of β (see next Section). For a fixed NT value and a small value of β, as we increase density the power consumption is dominated by the passive nodes in the passive-sleep cycle. The intuition is that the smaller the α, the larger the Ts in relation to Tp, and consequently, the larger the energy savings the system can achieve. Note that these savings come at a cost; the larger the Ts, the larger the reaction time of

[1] The difference in power consumption for Idle, Tx, and Rx is not significant (see Section IV.C).



(a) PC-104 Node (b) PC-104 DebugStation (c) RPC Radiometrix Radio

Fig. 6. Hardware Components.

the system in case of dynamics. Eq. (8) shows the upper bound of the energy savings as we increase density.

B. Goals and Metrics

Our goals in evaluating ASCENT were three-fold: First, in order to validate some of the assumptions made during design of the algorithm, perform simulations and real experiments and conduct comparative performance evaluation of the system with and without ASCENT. Second, understand the energy savings that can be obtained by using ASCENT. Finally, study the sensitivity of ASCENT performance to the choice of parameters.

We choose 4 metrics to analyze the performance of ASCENT: **Packet Loss** measures the percentage of packets not received by any node in the network[2]. When all the nodes are turned on –we call this the Active case– the packet loss includes all nodes. In the ASCENT case, it includes all nodes but the ones in the sleep state. This metric indicates the real bandwidth available to the nodes in the sensor network. **Event Delivery Ratio** is the ratio of the number of distinct event packets received by the sink to the number originally sent by the source. A similar metric has been used in ad-hoc routing [2]. **Energy Savings** is the ratio of the energy consumed by the Active case to the energy consumed by the ASCENT case. This metric defines the amount of energy savings we gain by using the ASCENT algorithm. **Network Lifetime** is defined as the time it takes for 90% of the transit nodes[3] to run out of energy. This metric provides an estimate of the longevity of the network.

C. Simulation and Experimental Methodology

Simulator: ASCENT was simulated using a high level, discrete event driven simulator. The simulator uses simulated time, not real time. Each node instance sets up its own stack, consisting of one or more modules. Nodes send events to a main event queue. Event messages are inserted in the event queue and removed from there during delivery. When removed from the queue, messages are first sent to a radio propagation module, and from there sent to the first module in the node's stack. Each module decides whether to continue the propagation of the message further up in the stack or not.

The simulator uses a simple propagation model. Transmissions with a distance between transmitter and receiver (T-R) within 0 to A range have probability 1 of being successful. Transmissions with a T-R within A to B have a linearly decreasing probability from 1 to 0 of being successful. Finally, transmissions with a T-R larger than B are not successful. Note that we chose a very simple propagation model for our simulator, since we complemented our simulations with **real experimental** results over **real propagation** channels.

Nodes have a collision module in their stack. This module does not propagate events further in the stack if two or more events are received within the contention period. Thus, the probability of collisions is determined by the length of the contention period and the number of neighbors forwarding messages.

Experimental Testbed: Fig. 6 shows pictures of the hardware components of our testbed. The basic node used in our testbed is the PC-104 node (Fig. 6(a)). It consists of a basic computing platform based on the PC-104 stack [19], and an RPC radio [23] (Fig. 6(c)). The DebugStations (Fig. 6(b)) are similar to our PC-104 nodes, except that they have an Ethernet connector, and 8-port serial port card, and they do not have RPC radios. The PC-104 nodes connect to the DebugStations through ppp-serial connections, and from there to Ethernet. This connection is used as an out of band channel for logging and management purposes. After connecting the DebugStations to Ethernet and the PC-104 nodes to the DebugStation through a serial connection, the *entire* process runs automatically. PC-104 nodes connect to the Master Debug Host at boot time to check if they need to update the software running. If they need it, they do the update and boot again. All the experiments' traces are reliably logged on the Master Debug Host for post- processing.

As we mentioned in section III.D, the ASCENT implementation runs on top of our radio MAC/device driver and below the diffusion routing daemon used (see Routing section below).

Scenarios and environment: In order to study the performance of ASCENT's algorithms as a function of density, we run experiments ranging from 5 to 25 nodes in increments of 5 nodes. One of the experiments consisted of placing 2 nodes near the maximum communication range from each other, and the rest of the nodes distributed between them. We did this for 2 reasons. First, we had a limited amount of nodes (30), and we wanted to stress ASCENT as much as possible by increasing local density. Second, we needed to validate our simulations with as high density as possible in order to further explore the scalability of our algorithm. We also ran some experiments with a multihop topology, albeit a small one. In addition to our limited node supply, the RPC radio range was

[2] Losses are detected using sequence numbers (Section III.C)

[3] Nodes that are in the transit path from the source to the sink.

around 10-15 meters. indoors with a 10dB attenuator (the range cannot be adjusted). This radio range made it difficult to effectively set up a large multi-hop topology due to the limited dimensions of our office environment. All the experiments were done in an indoors environment, with obstacles such as, furniture, walls, doors, etc. The simulations replicate the same scenarios tried in the experiments. For each simulation, we vary the density of nodes from 5 to 80 nodes. In addition, for larger multihop simulations, we incremented the number of sources and sinks from 1 to 5. In all the experiments and simulations, the source(s) and the sink(s) were placed at the edge of the network to maximize the number of hops and usage of transit nodes. Each experimental point in the graphs presented in the following sections is the average of three trials, and each simulation point in the graphs represents the average of 10 simulations (all of them with standard deviation).

Traffic: In each experiment, one source sends approximately 100 messages with temperature and light sensor readings (since the sensor card was not finished at the time the experiments were performed, the readings were stored values). The data rate was set to 3 sensor reading messages per minute, unless otherwise indicated. In each simulation, one or more sources send approximately 400 messages each. The data rate was 1 message every 200 units of simulation time. In all our experiments and simulations we operate the sensor network far from overload. Hence, our sensor nodes do not experience congestion. Understanding the performance implications of congestion on our algorithms is the subject of future research. In spite of experimenting with uncongested networks, our nodes can incur message losses due to dynamics and interference.

Packet fragmentation and randomization: The size of the packets used range from 100 to 150 bytes. The RPC maximum fragment size is 27 bytes, which implies that between 4 to 6 radio fragments are sent per packet. The loss of a fragment leads the loss of the entire packet (our MAC discards all the associated fragments received, there is no fragment retransmission mechanism). The RPC driver [24] implements a CSMA-style MAC with a simple collision avoidance mechanism. It does not emit packets until a (randomized) quiet interval has passed, i.e. an interval during which no packets have been received from other radios. We call this mandatory waiting time between receiving and sending the hold-off period. This simple scheme works because packet inter-arrival times are usually correlated due to fragmentation. Multiple nodes that are waiting to transmit will hopefully not collide due to the randomization of the hold-off period. In our experiments, the hold-off period was set to 100 ms and the randomization time to 200 ms. We set these values by carefully measuring the minimum time required by the RPC hardware and driver to transmit consecutive fragments. In our simulations we did not include any details regarding low-level fragmentation. Since at each hop multiple nodes may try to forward the same message, there is some contention for channel utilization. We provide application level randomization for packet forwarding that is set to a maximum of 10 seconds for all the experiments. We also added a similar level of randomization in our simulations.

Routing: We use directed diffusion as our routing protocol. Due to lack of space, we refer to [13] for a complete description of the protocol. In our implementation, the interest timeout was set to 100 sec., and the gradient timeout was set to 180 sec. Due to a limitation in our implementation, we were not able to turn off the radio and shut down the directed diffusion daemon when ASCENT was in the sleep state. Packets received while in sleep state were filtered and ignored as if they weren't received by ASCENT. This situation artificially inflated the packet losses experienced due to collisions of data from active nodes and diffusion control packets from sleep nodes. We did not consider these collisions in the results.

Energy Model: To model the energy consumption, we took measurements of the Radiometrix RPC RPS-418-40 radio [23]. In addition, we looked at the manual specifications of the RFM Tx-6000 [22], another low power radio used in our testbed. We found that the values for Tx:Rx:Idle:Sleep in mW were 75:100:100:0.4 for the RPC, and 40:40:36:0.015 for the RFM. Note that the Tx power of the RPC is smaller than the Rx and Idle power. We believe this is due to power consumption of the RPC's leds; the Tx led is significantly smaller than the other two. Several studies [4, 29] have reported differences of the order of 10:1 between Idle and Sleeping power consumption for 802.11 wireless LAN cards. For the low power radios we study, this difference is in the order of 100:1. This relation is important since it is the β factor defined in the previous section. In our model, we did not consider the energy consumed by the CPU.

The remainder of this section presents our simulation and experimental results.

D. Network Capacity

Our first simulations and experiments compare the packet loss rate and the event delivery ratio of the system with and without ASCENT. Fig. 7 shows the packet loss as a function of the density in a 2-hop network. We present analytical, simulation, and experimental results for the system with and without ASCENT. The results are encouraging. To a first degree, there are no important differences between the expected behavior and the real behavior up to 25 nodes, the maximum number of nodes in our experiments. In the Active case (no self-configuration, all nodes are turned on), all the nodes join the network and transmit packets. This case has high packet loss because as we increase the density of nodes, the probability of collisions increases accordingly. It rapidly reaches around 70% with 15 nodes, and enters into a saturation region after that. ASCENT limits the number of active nodes to the NT value, and therefore does not increase channel contention with larger densities. Fig. 8 shows the event delivery ratio, i.e. the percentage of events transmitted by the source that reached the sink. The experiments were done on a small multi-hop topology with a maximum density of 15 nodes per radio range (using all our nodes). On average, each packet traverses 3 hops. The simulations were done on a larger network, with packets traversing on average 6 hops from source to sink. We can see that ASCENT outperforms the Active case. ASCENT's performance remains stable as the density increases, which demonstrates the scalability properties of our algorithms as the number of nodes increases. The Active case does not perform as bad as one would expect based on the packet loss shown in the previous graph. This is because the event delivery ratio metric only requires that at least *one* copy

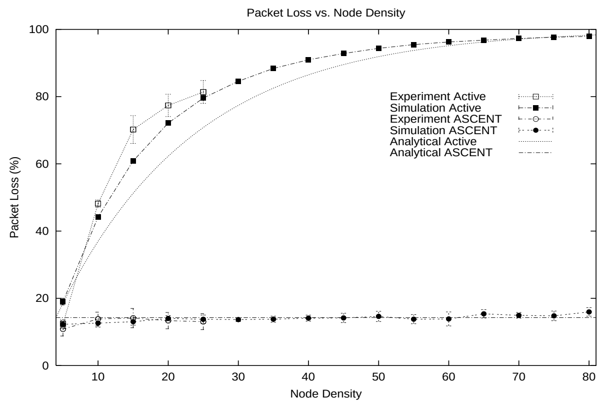


Fig. 7. Packet loss as a function of node density. Each packet traverses an average of two hops. ASCENT limits the number of active nodes to NT and reduces the contention for the channel.

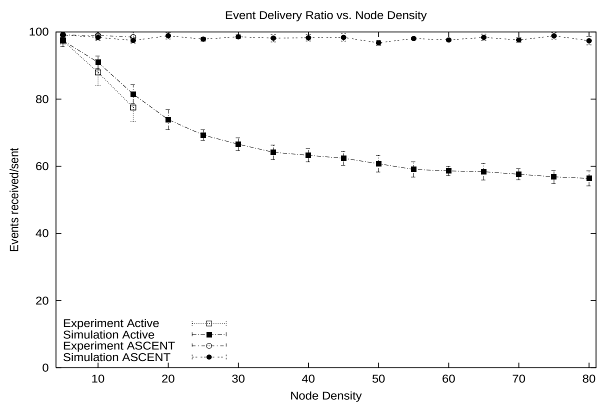


Fig. 8. Event delivery ratio as a function of node density. Each packet traverses three hops in the experiments and 6 hops in the simulations. ASCENT delivery rate is stable for the range of densities we tried.

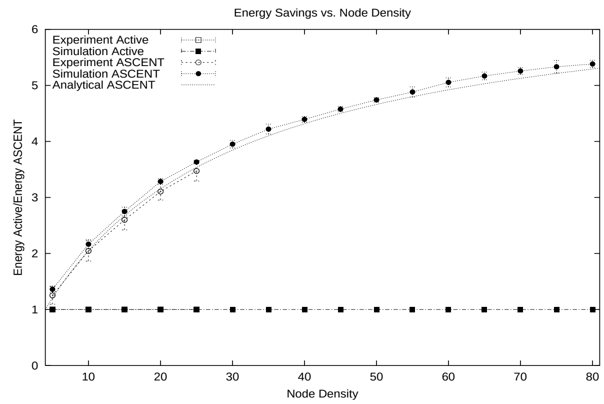


Fig. 9. Ratio of energy used by the Active case to the energy used by ASCENT. ASCENT provides significant amount of energy savings over the Active case.

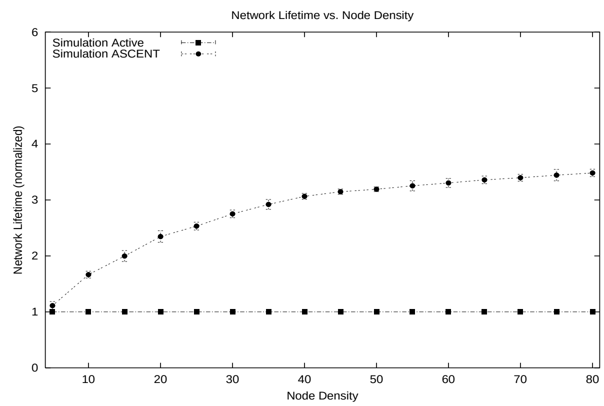


Fig. 10. Network lifetime, defined as the time it takes for 90% of the transit nodes to run out of energy, as a function of density. It is normalized to the lifetime of the Active case. ASCENT is a factor of 3 better than without.

of the original message sent by the source reach the sink. Even in a high-density environment with high losses, the likelihood of receiving *one* copy of the message is still high using diffusion routing [13].

E. Energy Savings

This section evaluates ASCENT's ability to save energy and increase network lifetime.

In these experiments and simulations, we did not consider the energy spent by the source(s) or the sink(s). For the experiments, the values are not direct measurements of energy consumption but indirect measurements using the time the nodes spent in the different ASCENT's states.

Fig. 9 shows the average energy consumption per node in a 2-hop network. From these results, we find that ASCENT provides a significant amount of energy savings over the Active case. We also find that as density increases, energy savings do not increase proportionally. This result may seem counterintuitive because in ASCENT the number of active nodes remains constant as density increases, and one would expect to save more energy as the fraction of active nodes decreases. From the analysis shown in Section IV.A, we see that the energy savings, as we increase density, depends on the passive-sleep cycle of the passive nodes, and not on the fraction of active nodes.

Fig. 10 shows the network lifetime as a function of density. The figure is normalized to the lifetime of the Active case. The results include simulations of a multihop network with an average hop count of 6. The results do not include experiments. Network lifetime with ASCENT is always larger than without it, and for large densities it is a factor of 3 better.

F. Impact of Various Factors

To explain how ASCENT reacts to different configuration parameters, we conducted sensitivity experiments and simulations on two parameters, randomization and data rate.

Fig. 11 shows the packet loss as a function of density for different randomization values. We clearly see that for larger randomization values we get fewer packet losses. However, there is a trade-off since larger randomization values increase the end-to-end delay for data delivery. For the different levels of randomization we tried, ASCENT case always outperforms the Active case, even when the former has less randomization than the latter.

Fig. 12 shows the packet loss as a function of the data rate for different experiments and simulations with density of 20 nodes. For all the cases we see that as we increase the data rate, the packet loss increases accordingly, albeit more slowly for ASCENT.

G. Discussion

Finding the right level of abstraction when performing a simulation is always a difficult problem. Too little detail may lead to misleading or incorrect results, and too much detail increases the development time, and the probability of errors [12]. In our case, having an experimental platform to run experiments and validate our simulations proved to be extremely valuable. In some cases we were correct in making certain simplifications to our simulation. For instance, we did not simulate the low level packet processing details, such as, fragmentation, MAC level collision avoidance mechanisms, and low-level timers. Our simple collision module, based on a configurable collision window, proved to be good enough. For the scenarios and level of dynamics we tried, the differences in the simulation and experimental results were not significant. In some other cases, the simplifications assumed in simulations did not reflect the reality. One observation from the experiments is that nodes that were geographically distant from a forwarder node, could in fact help and relay packets to nodes that were geographically closer to the forwarder. Even though this fact did not change our results (since ASCENT is based on connectivity, not on geographical position), this situation never happened in our simulations. This is an artifact of our simple propagation model. We believe that better propagation models, like Log-normal Shadowing, could improve this situation and we plan to run further simulations with more complex propagation models. Nevertheless, from our experiments in indoors environments, we detected the occurrence of blind spots that are not easily captured by probabilistic propagation models. This fact reinforces the viability of our design in practice. Distributed algorithms for wireless sensor networks *must* adapt and self-configure to the conditions *measured locally*. We believe that schemes based on geographical proximity or assuming certain propagation conditions may not work in practice for indoors environments.

V. RELATED WORK

Our work has been informed and influenced by a variety of other research efforts.

K. Sohrabi and G. Pottie [28] have made significant progress in self-configuration and synchronization in sensor networks at the single cluster level with a TDMA scheme. This work shares with us similar design principles, although it's more focused on low-level synchronization necessary for network self-assembly, while we concentrate on efficient multi-hop topology formation. J. L. Gao's thesis [11] presented an adaptive local network formation/routing algorithm that facilitates cooperative signal processing. An election algorithm is used to select a central node among a small group of nodes that cooperate in information processing. While these algorithms were designed to operate for a relatively short time span in a reduced area near the target event, our objective is stable, long range topology formation that covers the entire sensor network.

The adaptive techniques we use were studied extensively to make the MAC layer self-configuring and adaptive more than 20 years ago during the refinement of contention protocols [14, 17]. More recently SRM [10] and RTCP [27] borrowed these techniques to adaptively adjust parameters such as session message frequency and randomization intervals. In this work we use those techniques to adapt the topology of a multi-hop wireless network.

Mobile ad-hoc networks [15, 18, 20] and directed diffusion [13] adaptively configure the routing or data dissemination paths, but they do not adapt the basic topology. Q. Li and D. Rus [16] presented a scheme where mobile nodes modify their trajectory to transmit messages in the context of disconnected

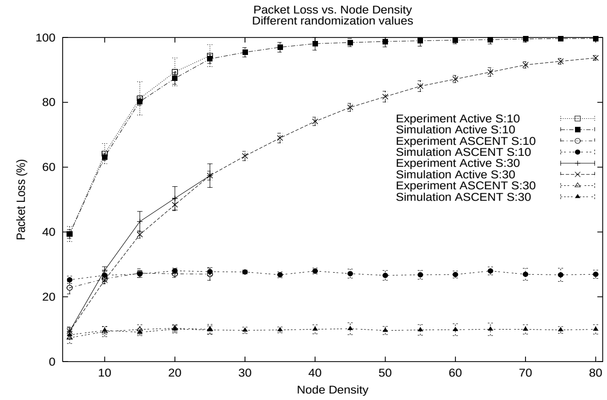


Fig. 11. Effect of randomization on packet loss as a function of density. The system always gets fewer losses with ASCENT than without it for any randomization level.

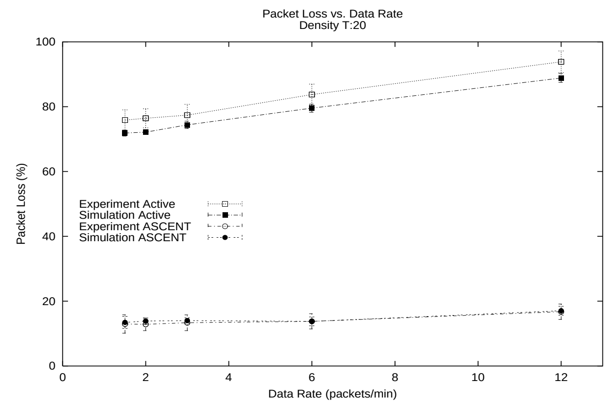


Fig. 12. Packet loss as a function of data rate. The losses tend to increase for larger data rates, although they do it more slowly for ASCENT.

ad-hoc networks. This work shares with us the notion of adaptation of the basic topology for efficient delivery of messages, but it does so by sending location updates between neighbors and using active messages to incrementally propagate them toward the destination. Our work uses measurements of neighbor density and packet loss to exploit the redundancy of dense areas in the system in an energy efficient way. This work may complement ours in case of mobile nodes deployment and in the presence of network partitions. Ramanathan et. al. [25] proposed some distributed heuristics to adaptively adjust node transmit powers in response to topological changes caused by mobile nodes. This work assumes that a routing protocol is running at all times and provides basic neighbor information that is used to dynamically adjust transmit power. In our case, ASCENT decides which nodes should run the routing algorithm, and it makes this determination based on packet loss in addition to density.

In Y. Xu et al. GAF [30], nodes use geographic location information to divide the network into fixed square grids. Nodes in each grid alternate between sleeping and listening, and there is always one node active to route packets per grid. ASCENT does not need any location aids, since it is based on connectivity. In addition, geographic proximity may not always lead to radio connectivity; this is why ASCENT uses local connectivity measurements. B. Chen et al. [4] proposed SPAN, an energy efficient algorithm for topology maintenance, where nodes decide whether to sleep or join the backbone based on connectivity information supplied by a routing protocol. ASCENT does not depend on routing information, nor needs to modify the routing state; it decides whether to join the network or sleep based on *measured* local connectivity and packet loss information. In addition, our work does not presume a particular model of fairness or network capacity that the application requires.

Self-configuration based on local measured parameters takes some inspiration from biological systems, in particular the models of ant colony behavior [5]. Bulusu et. al. [3], have proposed different algorithms for incremental beacon placement in sensor networks. This work share with us the same design principles, such as the use of localized algorithms, and adaptation based on locally measured parameters. While their work is oriented to solve the localization problem, ours is more oriented to energy efficient communication and sensing coverage.

VI. Conclusions and Future Work

In this paper, we described the design, implementation, analysis, simulation, and experimental evaluation of ASCENT, an adaptive self-configuration topology mechanism for distributed wireless sensor networks. There are many lessons we can draw from our preliminary experimentation. First, ASCENT has the potential for significant reduction of packet loss and increase in energy efficiency. Second, ASCENT mechanisms were responsive and stable under systematically varied conditions.

In the near future, we will perform experiments with larger numbers of nodes to further explore the scalability of our algorithms. We will investigate the use of load balancing techniques to distribute the energy load, and explore the use of wider area links to detect network partitions. We will also expand this work to address other modalities beyond communication and sensing coverage, such as, actuation.

This work is an initial foray into the design of self-configuring mechanisms for wireless sensor networks. Our distributed sensing network simulations and experiments represent a non-trivial exploration of the problem space. Such techniques will find increasing importance as the community seeks ways to exploit the redundancy offered by cheap, widely available microsensors, as a way of addressing new dimensions of network performance such as network-lifetime.

Acknowledgments

This work is supported by NSF under grant ANI-9979457 as the SCOWR project. The authors would like to acknowledge the discussions and suggestions from members of the SCOWR project and the UCLA LECS Lab. We would like to specially thank Jerry Zhao for being so helpful with testbed issues, Jeremy Elson for writing the RPC driver, Fabio Silva for providing the diffusion implementation, and David Braginsky for writing the simulator used in this work. Finally, the authors would like to thank John Heidemann, Ramesh Govindan and the anonymous reviewers who made valuable comments that helped improve previous versions of this paper.

References

[1] B. Badrinath, M. Srivastava, K.Mills, J.Scholtz, and K.Sollins, Eds. Special Issue on Smart Spaces and Environments. IEEE Personal Communications, Oct. 2000.

[2] J. Broch, D. Maltz, D. Johnson, Y.-C. Hu, and J. Jetcheva. A Performance Comparison of Multi-Hop Wireless Ad-Hic Network Routing Protocols. In Proceedings of Mobicom'98, Dallas, TX, 1998.

[3] N. Bulusu, J. Heidemann and D. Estrin. Adaptive Beacon Placement. In Proceedings ICDCS-21, Phoenix, Arizona, USA. April 2001.

[4] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris. Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks. In Proceedings Mobicom '01, Rome, Italy, July 2001.

[5] G. Di Caro and M. Dorigo. AntNet: A Mobile Agents Approach to Adaptive Routing. Technical Report 97-12, IRIDIA, Universite' Libre de Bruxelles, 1997.

[6] J. Elson and D. Estrin. Random, Ephemeral Transaction Identifiers in Dynamic Sensor Networks. In Proceedings of ICDCS-21, Phoenix, Arizona, April 2001.

[7] D. Estrin, R. Govindan, and J. Heidemann, Eds. Special Issue on Embedding the Internet. Comm. of the ACM, vol. 43, no. 5, May 2000.

[8] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar. Next century challenges: Scalable Coordination in Sensor Networks. In Proceedings of Mobicom '99, Seattle, Washington, August, 1999.

[9] W. Fenner. Internet Group Management Protocol, Version 2. RFC-2236, November 1997.

[10] S. Floyd, V. Jacobson, C-G. Liu, S. McCanne, and L. Zhang. A Reliable Multicast Framework for Lightweight Sessions and Application Level Framing. IEEE/ACM Transactions on Networking, November 1997.

[11] J. Gao. Energy Efficient Routing for Wireless Sensor Networks. PhD thesis in Electrical Engineering, UCLA, August 2000.

[12] J. Heidemann, N. Bulusu, J. Elson, C. Intanagonwiwat, K. Lan, Y. Xu, W. Ye, D. Estrin, and R. Govindan. Effects of Detail in Wireless Network Simulation. In Proceedings of the SCS Multiconference on Distributed Simulation, pp. 3-11. Phoenix, Arizona, January, 2001.

[13] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks. In Proceedings of Mobicom '00, Boston, Massachusetts, August 2000.

[14] V. Jacobson. Congestion Avoidance and Control. Proceedings of SIGCOMM '88, pages 314-329. Palo Alto, CA, August 1988.

[15] D. Johnson and D. Maltz. Dynamic Source Routing in Ad-hoc Wireless Networks. In T. Imielinski and H Korth, editors, Mobile Computing, pages 153-181. Kluwer Academic Publishers, 1996.

[16] Q. Li and D. Rus. Sending Messages to Mobile Users in Disconnected Ad-hoc Wireless Networks. In Proceedings of Mobicom '00, pages 44-55, Boston, August 2000.

[17] R. Metcalfe and D. Boggs. Ethernet: Distributed Packet Switching for Local Computer Networks. Communications of the ACM, 19 (5): 395-404, July 1976.

[18] V. Park and M. Corson. A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks. In Proceedings of Infocom '97, pages 1405-1414, April 1997

[19] PC-104 Consortium, http://www.pc104.org

[20] C. Perkins and E. Royer. Ad hoc On-Demand Distance Vector Routing. Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications, New Orleans, LA, February 1999, pp. 90-100.

[21] G. Pottie and W. Kaiser. Wireless Integrated Network Sensors. Communications of the ACM, 43 (5): 51-58, May 2000.

[22] RFM TX6000 hybrid transmitter, http://www.rfm.com

[23] Radio Packet Controller, http://www.radiometrix.com

[24] RPC driver, http://www.circlemud.org/~jelson/software/radiometrix

[25] S. Ramanathan and R. Rosales-Hain. Topology Control of Multihop Radio Networks using Transmit Power Adjustment. In Proceedings of IEEE Infocom '00, Tel Aviv, Mar 2000.

[26] Sensors: The Journal of Applied Sensing Technology.

[27] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RFC 1889 RTP: A Transport Protocol for Real-Time Applications, January 1996.

[28] K. Sohrabi and G. Pottie. Performance of a Novel Self-Organization Protocol for Wireless Ad-Hoc Sensor Networks. In Proceedings of IEEE VTC, Amsterdam, Netherlands, September 1999.

[29] M. Stemm and R. H. Katz. Measuring and reducing energy consumption of network interfaces in hand-held devices. IEICE Transactions on Communications, E80-B(8):1125-1131, Aug 1997.

[30] Y. Xu, J. Heidemann, and D. Estrin. Geography-informed Energy Conservation for Ad Hoc Routing. In Proceedings of MobiCom '01, Rome, Italy, July 16-21, 2001.

[31] W. Ye, J. Heidemann, D. Estrin. An Energy-Efficient MAC Protocol for Wireless Sensor Networks. In Proceedings of IEEE Infocom '02, New York, New York, June 23-27, 2002.