ARRIVE: Algorithm for Robust Routing in Volatile Environments

Chris Karlof Yaping Li Joseph Polastre
{ckarlof,yaping,polastre}@cs.berkeley.edu
Department of EECS, University of California at Berkeley

***Abstract*— Wireless sensor networks for environmental monitoring and distributed control will be deployed on a large scale in the near future. Due to the low per-node cost, these networks are expected to be both large and dense. However, because of the limited computation, storage, and power available to each node, conventional ad-hoc routing techniques are not feasible in this domain, and more novel routing algorithms are required. Despite the need for a simpler approach, routing in sensor networks still needs to be both robust to failures and secure against compromised and malicious nodes. We propose ARRIVE, a probabilistic algorithm that leverages the high node density and the inherent broadcast medium found in sensor networks to achieve routing robust to both link failures and patterned node failures without resorting to periodic flooding of the network. Our algorithm is based on a tree-like topology rooted at the sink of the network, and nodes use localized observed behavior of the surrounding nodes to make probabilistic decisions for forwarding packets. We have found that ARRIVE adapts to large patterned failures within a relatively short period of time at the cost of only moderate increases in overall power consumption and source-to-sink latency.**

I. Introduction

Wireless sensor networks are currently an active area of research for a variety of topics ranging from distributed control of micro-mechanical actuators to database query processing. Foreseeable applications involving sensor networks include environmental and medical monitoring, energy management, inventory control, and pursuit/evader games in a battlefield setting [1]. Nodes in sensor networks have extremely constrained computational, storage, and energy resources, and this limits the types of routing mechanisms that can efficiently be deployed. Furthermore, conventional ad-hoc routing algorithms for wireless networks typically support IP-style addressing of destinations [2], [3], [4] and use intermediate nodes to support general purpose end-to-end connections between arbitrary pairs of nodes within the network. Although it is possible that any-to-any communication might be still be relevant in a sensor network, other communication paradigms are more prevalent.

Sensor networks often have one or more points of centralized control called *sinks*. A sink is typically a gateway to another network, a powerful data processing or storage center, or an access point for human interface. They can be used as points to disseminate control information into the network or extract data from it. In environmental sensing applications, sinks can also be used as gateways to issue queries to *source* nodes, requesting sensory information such as light or temperature, or posting interests to be notified of the occurrence of events like movement or some unusual activity. In this context, many-to-one and one-to-many communication patterns are more probable than any-to-any.

Many routing mechanisms for sensor networks have been proposed in the literature. Some are single path algorithms [5], [6] which rely on sinks to periodically flood the network to discover new routes to redirect traffic around failed or malicious nodes. Flooding the network is a highly expensive operation with respect to energy consumption and should be avoided. Furthermore, during the period between flooding, these algorithms offer no mechanism to route around failed or malicious nodes. Location-aware routing algorithms [7], [8] offer interesting solutions, however they require every node is able to determine its geographic position. Localization can be done either with GPS or a distributed algorithm, but both can carry significant costs in terms of power consumption, specialized hardware, or algorithmic complexity. Furthermore, it may not be feasible for all types of networks (GPS will not work in indoors or in outer space), and the resulting routes still tend to be single path.

We propose ARRIVE, a probabilistic routing algorithm that relies only on localized information to make packet forwarding decisions. ARRIVE provides substantial end-to-end reliability and is resilient to three sources of packet

loss expected to be found in sensor networks: isolated link failure, patterned node failures, and malicious or misbehaving nodes. Conventional protocols for achieving end-to-end reliability like TCP [9] rely on explicit acknowledgments in the transport layer and assume that network congestion is the dominant cause of packet loss. Loss or corruption at the physical and link layers or outright node failure is a more prevalent detriment to end-to-end reliability in sensor networks, and retransmission over a lossy path is unlikely to be beneficial. In addition to the issues of increased energy consumption and bandwidth usage caused by end-to-end acknowledgments, the high loss rates found in sensor networks can also prove troublesome. Acknowledgment packets are likely to be lost as well, causing a vicious cycle of retransmissions on both ends and a dramatic reduction in throughput.

In contrast, ARRIVE uses a *forward* approach to end-to-end reliability. Whereas conventional algorithms *detect* packet loss through explicit acknowledgments, ARRIVE *avoids* packet loss by sending multiple packets representing a single event or data reading. The intuition is obvious: since a single packet is highly prone to loss within the network, sending multiple packets will likely increase the chance of reception by the sink node. As mentioned above, this must be done with caution. If there is a single bad link that is causing the loss, any number of packets sent over the same bad link will also be lost. ARRIVE promotes *diversity* in paths in two fundamentals ways:

1) When a node receives a packet, the next hop is chosen probabilistically based on link reliability and node reputation. Each packet takes a loosely controlled random walk between the source and sink, collectively forming a *beam*.
2) When a node processes two or more packets representing the same event, it ensures these packets use different outgoing links. The benefits of sourcing multiple packets is lost if they are all sent over the same weak link or to the same misbehaving node. Detecting these *convergences* helps prevent multiple packets from encountering the same source of failure.

ARRIVE takes advantage of the inherent broadcast medium through *passive participation*. If Node A observes Node B sent a packet to Node C, but Node A does not overhear Nodes C forward the packet to another destination, with some small probability Node A will take responsibility for forwarding the packet itself. This technique can also be used as a defense against malicious nodes. Passive participation must be used with caution. Unidirectional links and the hidden terminal problem may cause the passively participating node to act erroneously, unnecessarily increasing the traffic in the network.

We make no special assumptions about the capabilities of the nodes in the implementation of our algorithm: localization is not required, storage requirements for local state at each node is minimal, and no specialized hardware is needed. We do assume that network is dense enough such that there is sufficient multiplicity of paths between the sources and sink for our algorithm to be effective. Sensors are expected to have a low per-node cost, so we believe this assumption to be reasonable.

We will show that ARRIVE improves reliability significantly in the presence of weak links and nodes failures. We use four metrics to evaluate our approach. *Event delivery ratio* is measured as the ratio of events received by the sink to events sourced. An event is an abstraction that represents any information originating from a source. An event could be a single sensor reading, the aggregation of readings from several neighboring nodes, or any other interesting information. The remaining three metrics explore the costs of deploying ARRIVE. Sourcing multiple packets corresponding to a single event has obvious costs of increased *bandwidth* and *energy* consumption. We will show how ARRIVE limits bandwidth consumption and examine the energy/reliability tradeoff in sourcing multiple packets. Finally, due to the probabilistic nature of our algorithm, the routes used by packets are unlikely to be optimal. We will show that the *additional hops* taken by packets is proportional to the distance(in hops) between the source and the sink.

II. Related Work

A. Routing algorithms

The design and analysis of ad-hoc routing protocols has been a rich area of research over the past seven years. Routing algorithms can be classified as *a priori* or *on-demand*. A priori protocols actively pre-compute routes in the network, even when no traffic is using them, while on-demand protocols will only discover routes when they are needed. It is unclear which of these approaches is best suited for sensor network traffic. A priori protocols have the expense of continual route maintenance, which can be a drain on the network's energy resources if no traffic is using the discovered routes. On-demand protocols typical have two phases: when a route is needed, it is first discovered and then taken. Data traffic is not sent along a route until the discovery stage is complete. Due to the high loss rate in sensor networks, packets containing discovered routes may be lost and never become known by the source node. For sensor networks, a promising approach is the combination of these two approaches, but where

route discovery occurs concurrently with data delivery. Popular ad-hoc routing algorithms include Destination-Sequenced Distance Vector Protocol (DSDV) [4], Ad-hoc On-demand Distance Vector Routing (AODV) [10], Dynamic Source Routing (DSR) [3] and the Temporally-Ordered Routing Algorithm (TORA) [2]. For the above mentioned reasons, these algorithms are generally unsuitable for large-scale, dense sensor networks.

Location-aware routing [7], [8] is promising for sensor networks, but it requires that each node knows its geographic location. GPS is usually needed for at least some of the nodes. GPS requires extra power and additional hardware, and doesn't work indoors.

Directed diffusion [5] is the process of registering an interest in the network and then reinforcing high bandwidth stable links. A sink will flood the network with an interest consisting of key-value pairs. At each hop, the network will set a gradient back to the sink. Once the sources receive the interest, they will begin to diffuse data to the sink via the gradients. As a path is used, a history of its characteristics is maintained. The paths with the best characteristics are reinforced. If a node on a path was to fail, data would be rerouted through the other existing gradients, and another path would be reinforced. The resulting routes are single-path, although there is further work exploring multi-path routes using directed diffusion [11].

TinyOS [6] is an event-driven operating system for tiny networked sensors. TinyOS currently runs on the Berkeley MICA platform (referred to as "motes") as well as a variety of other previous generation motes. Programming applications for TinyOS is done in a C-like environment with the flexibility to interface with any of the mote's hardware components.

Current routing implementations in TinyOS are naive and unreliable. The `router` component developed by J. Hill and J. Polastre floods the network with a packet from the sink, and each mote determines its level in the network and sets its parent to the mote it can hear that is closest to the sink. Packets are then routed via the parent points. BLESS, a beacon-less protocol developed by P. Levis, listens to sensor data being broadcast on the network and uses this data to flood messages back to the base station.

SPAN [12] and GAF [13] are two algorithms that attempt to reduce energy consumption in ad-hoc wireless networks by creating a sleep schedule for nodes in a way such that routing fidelity is not sacrificed. These techniques are not a routing algorithm in themselves, but are intended to complement existing routing infrastructure.

B. Security

One of the design goals of ARRIVE is to be secure against malicious and misbehaving nodes. Security in ad-hoc wireless networks is substantially different from the wired world, but the underlying issues are the same.

Stajano and Anderson [14] point out that traditionally, security concerns are usually addressed in the following order: confidentially, integrity, authenticity, and availability. But in ad-hoc networks, their relative importance is often in the opposite order. This is particularly true for sensor networks. Wireless sensors are meant to be cheap and deployed in large amounts, and possibly simply abandoned when they run out of power. Thus, it is reasonable to assume an adversary could obtain some "real" nodes, read off any cryptographic information, download some malicious code, and re-introduce them to the network. Confidentiality becomes a secondary issue when it is difficult to verify the authenticity of the sender or receiver. An even more pressing issue is the availability of the network. Since malicious nodes can be active participants, one goal of such nodes may be to disable the network, in which the issues of confidentiality and integrity become irrelevant.

Public key cryptography is a obvious candidate for security in sensor networks, however it is beyond the computational power of the the motes and unreasonable for each mote to store the public keys of all the nodes in the network. Efficient mechanisms for data confidentiality, two-way authentication, and data freshness using only symmetric key cryptographic primitives in sensor networks are explored by Perrig et al. [1]. These methods are useful in securing networks against external attackers, but are less effective in the presence of compromised nodes.

C. Passive participation

Aron and Gupta introduce the concept of *witness hosts* to provide reliability in the presence of unidirectional or failed links [15]. The WAR (witness aided routing) protocol instructs witness nodes to bypass a faulty or unidirectional link.

Marti et al. examine techniques to improve throughput in ad-hoc networks in the presence of misbehaving nodes [16]. Their work takes advantage of the fact that nodes in ad-hoc wireless networks can *overhear* each other's transmissions and uses the good nodes to try to detect when bad nodes are supposed to forward packets but fail to do so. A "blacklist" of bad nodes is then propagated throughout the network. Their solution is dependent on Dynamic Source Routing, and has the problem that it is possible for nodes to blackmail each other, devaluing the entire system.

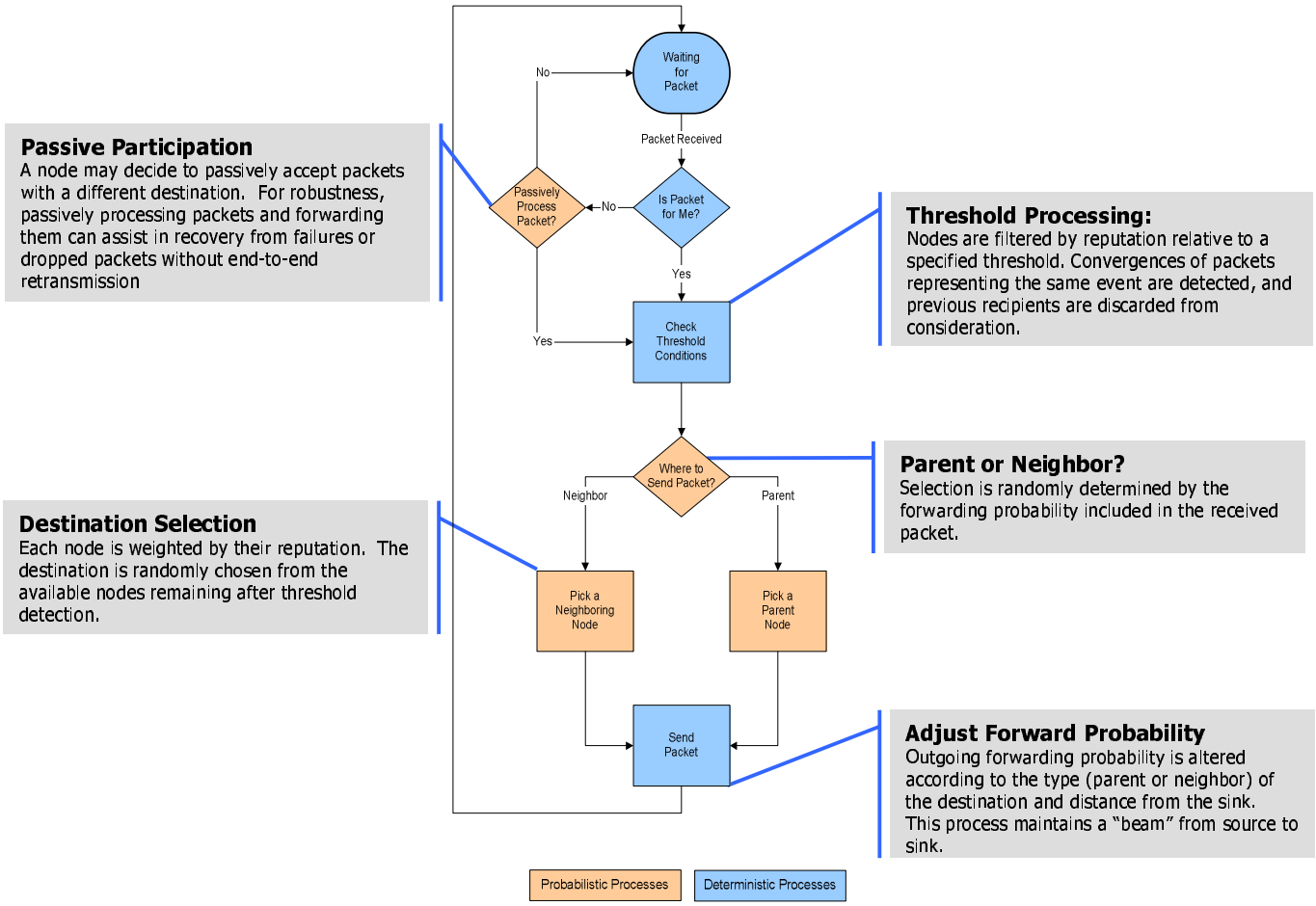

Fig. 1. Overview of ARRIVE

III. Algorithm

A. Terminology

Below we define the terms that we use in describing ARRIVE.

Event: An event could be a single sensor reading, the aggregation of readings form several neighboring nodes, or any other interesting information. Events are uniquely identified by a [Source ID,Event ID] pair. Events always result in a sequence of data that must be relayed to the sink, or data-collection point.

Level: Each node has a unique level number which indicates the distance (in hops) it is from the sink. The level of a sink is 0. All nodes that can hear the sink are at level 1. All nodes that can hear a level-1 node and that are not at levels lower than 2 are level-2 nodes, and so on.

Parents: The parents of a node N are the nodes that are at one level closer to the sink than N and that N can hear.

Neighbors: The neighbors of a node N are nodes that are at the same level as N and that N can hear.

Push: We say a node N pushes a packet P if N sends P to one of its neighbors.

Forward: We say a node N forwards a packet P if N sends P to one of its parents.

Forwarding Probability: We include a forwarding probability in the packet header for the proper recipient to probabilistically choose whether to push or forward the packet on the next hop.

A Reputation History of Node P with respect to Node N: Each node keeps a reputation history for each of its parents and neighbors. A history for a Node P contains C reputation periods. Every period is T seconds long. For each period, a number S represents the number of packets N sends to P and a number R represents the number of

packets N overhears P relaying during that period. R includes only the packets that were sent from N. This serves as an implicit acknowledgement. When a period expires, the oldest period in the reputation history is discarded, and tallies for S and R are reset to 0 for the upcoming period.

Reputation of Node P with respect to Node N: This is the ratio between the sum of the R's, the number of packets N has sent to P, and the sum of the S's, the number of packets relayed by P, over all periods. Each period is weighted with an exponential decay such that more recent reputation periods have greater weight.

Destination: We use destination to indicate the next hop.

Fanout: The number of packets sent by a source node corresponding to a unique [Source ID, Event ID] pair.

Convergence: Two packets representing the same event E received by a node N are considered a convergence of E at node N.

Convergence History at Node N of Event E: For each distinct event E, node N keeps a history of all destinations to which N sent the packet(s) representing E.

B. Algorithm Description

Each node keeps the following state information: level, parents, neighbors, reputation history, and convergence history. We use a breadth first search beaconing algorithm rooted at the sink to initialize the level, parents, and neighbors state information in each node.

We illustrate our routing algorithm with the flow chart in Figure 1. Whenever a node N hears a packet, it first checks to see if the packet is addressed to it. If it is, then N filters the following nodes from consideration for the next hop:

1) N does threshold detection processing to exclude parents and neighbors whose reputations are below a predefined threshold value.
2) N excludes all nodes that are in the convergence history of the event represented by the received packet.
3) N excludes the previous sender to prevent an immediate cycle.

According to the forwarding probability, N chooses probabilistically between its parents and neighbors to send the packet. Based on the result, N selects a parent or neighbor probabilistically weighted by the reputations of all remaining parents/neighbors. When pushing, the forwarding probability $Pr[f]$ is adjusted using the following formulas.

$$\mathbf{Pr}[f] = \mathbf{Pr}[f] + \left(\frac{1}{level} \times (1 - \mathbf{Pr}[f])\right)$$

$\mathbf{Pr}[f]$ remains the same when forwarding a packet. After adjusting the probability, N sends the packet.

With some low probability, $\mathbf{Pr}$[passive participation], nodes may decide to process and forward packets not destined for it. One condition in which a node N may passively participate is when N overhears a packet destined for a node P within its radio range, but then does not hear P attempt to forward that packet. N will then take the responsibility of forwarding that packet itself, assuming either P is malicious, dead, or on the receiving end of a bad link. Due to the possibility of unidirectional links in sensor networks, P may have actually forwarded the packet, and N simply did not hear it, unnecessarily increasing the traffic in the network.

The other condition when a node N may passive participate is when N overhears a packet destined for a node P within its radio range, and P attempts to forward the packet to another node Q not in N's radio range. This mechanism is intended for hostile environments with a high number of compromised nodes. In order to artificially increase one's reputation, a malicious node may forward received packets to non-existent nodes. This use of passive participation is a possible defense against such an attack. The downside is an unnecessary abundance of additional packets due to the well-known *hidden terminal* problem [17]. It is possible that a friendly node P simply forwarded the packet to a node outside of N's radio range.

IV. Methodology

In order to determine the effectiveness of ARRIVE, we developed two simulation environments. The goal of these environments is to model an actual sensor network architecture while being able to control the environment to study the reactivity of our algorithm.

The reference platform for ARRIVE is the Berkeley MICA hardware platform [18]. The platform features an Atmel Atmega103L processor with 128 kilobytes of instruction memory and an RFM TR1000 radio operating at 40 kilobits per second using amplitude shift keying. Previous research has shown that node-to-node communication in sensor networks is unreliable [19]. As a result, we've designed ARRIVE and simulated its performance in environments where the reliability of single hop communication may be 90% or lower.

To effectively test each of the parameters of ARRIVE, a simple Java simulation runs the algorithm with set parameters and no radio model. Using this simulator, we were able to debug our algorithm, test boundary conditions, and evaluate the performance of interchanging different subcomponents. This simulator was a quick method to attain results and feedback. We were able to run a statisti-

cally significant amount of simulations in a short amount of time; this is in contrast to larger, more encompassing simulators like ns-2.

A second simulator, the Berkeley ns-2 simulator [20], emulates the actual network environment including radio propagation model and MAC layer. Using ns-2, we can simulate more accurately what will happen when our algorithm is deployed on the MICA platform. We used the 802.11 MAC layer implemented in ns-2. This is not a satisfactory choice for a MAC, but we altered it several ways to make it more realistic. First, we lowered the bandwidth from 1.6Mbs to 40kbs match the RFM TR1000 radio. Secondly, 802.11 uses link layer acknowledgments to improve link reliability. Given no such mechanism currently exists in sensor networks, every packet was sent in "broadcast" mode, which disables these acknowledgments. Looking to the future, a MAC layer using channel acquisition based on RTS/CTS like 802.11 is not optimal for sensor networks. A TDMA-style MAC is more appropriate; it allows nodes to put their radios in standby node during idle periods [5]. Ns-2 also allowed us to measure power consumption in the network. To mimic the RFM TR1000, we altered the transmit and receive power consumption to 39mW and 13 mW. Nodes consumed no power while idle. This is not realistic in sensor networks (and ad-hoc wireless networks in general) since about the same amount of energy is dissipated listening for a packet as while receiving one [21]. However, since our goal is to analyze the energy tradeoffs of sourcing multiple packets, having nodes consume energy while listening but not receiving would cause idle energy consumption to dominate and make our results hard to interpret. Overall, we feel confident that the changes made to the 802.11 MAC layer allowed us to realistically model current sensor network technology. Ns-2 was primarily used to analyze ARRIVE's performance in the presence of patterned node failures.

Both simulators take density, link reliability, and number of sourced packets as a parameter. The density determines the size of the local neighborhood of each node. We divide a 1000 by 1000 unit grid into 100 boxes of size 100 by 100 units. For density δ, we place δ nodes uniformly at random at locations in each box for a total of $\delta \times 100$ nodes in the graph. The connectivity of the graph is determined by a fixed transmission radius of 75 units. We place the sink in the center of the graph. Nodes that source packets are chosen randomly from a distance of 10 levels away from the sink, as shown in Figure 2(a). This ensures that our results are representative of a long multi-hop path from source to sink. It also permits the introduction of failures at various distances from the source. A single path from the sink to the source can be seen in Figure 2(b) where the connectivity radius defines the connectivity of each node.

We assume that nodes are stationary throughout the simulation. If there is significant movement, state information can be updated with periodic flooding. Flooding, however, is something we want to avoid, and mechanisms to efficiently update level, parent, and neighbor information without flooding the network we leave as an important area of future work.

Changing the link reliability shows the adaptation of the algorithm to various volatile environments. The Java simulator tests links that are 90% reliable. Ns-2, in testing failures, assumes 100% link reliability–thus all lost packets are due to sending to a failed node.

Failures are introduced into the network after 50 seconds of the simulation have completed. Waiting is essential so that each node may build up reputations about its neighbors. The failure is circular in shape and is placed directly in the path between the source and sink. The number of nodes affected by the failure ranges from 0 to 100 in increments of 10. Figure 2(c) illustrates a failure introduced into the network between the source and sink. A path around the failure that may be probabilistically chosen by ARRIVE is shown to the right of the failure.

We experimented with failures at two different levels in the network. In the first experiment, the failure was centered 3 hops from the sink, and in the second experiment it was 5. It is intuitive that failures near the base station would have a greater effect on overall reliability, and we want to study how dramatic of an effect this will have on our algorithm.

V. Analysis

A. Parameter Analysis

We first wanted to understand the effectiveness of sourcing multiple packets when a given event occurs. We show in Figure 3 the effect of sourcing multiple packets with our algorithm over various densities. When only one packet is sourced, the results are dismal–over all densities, the number of events the sink receives is only about 28%. We expect the reliability to be constant across density since relying on any one path is equally bad. However the results are significantly more encouraging as four, six, and eight packets are sourced per event. The reliability increases with density since more destination choices at each hop leads to packets that are more likely to take independent paths.

A major concern of sending multiple packets per event in a sensor network is that these packets will lead to significantly increased bandwidth consumption. Increased

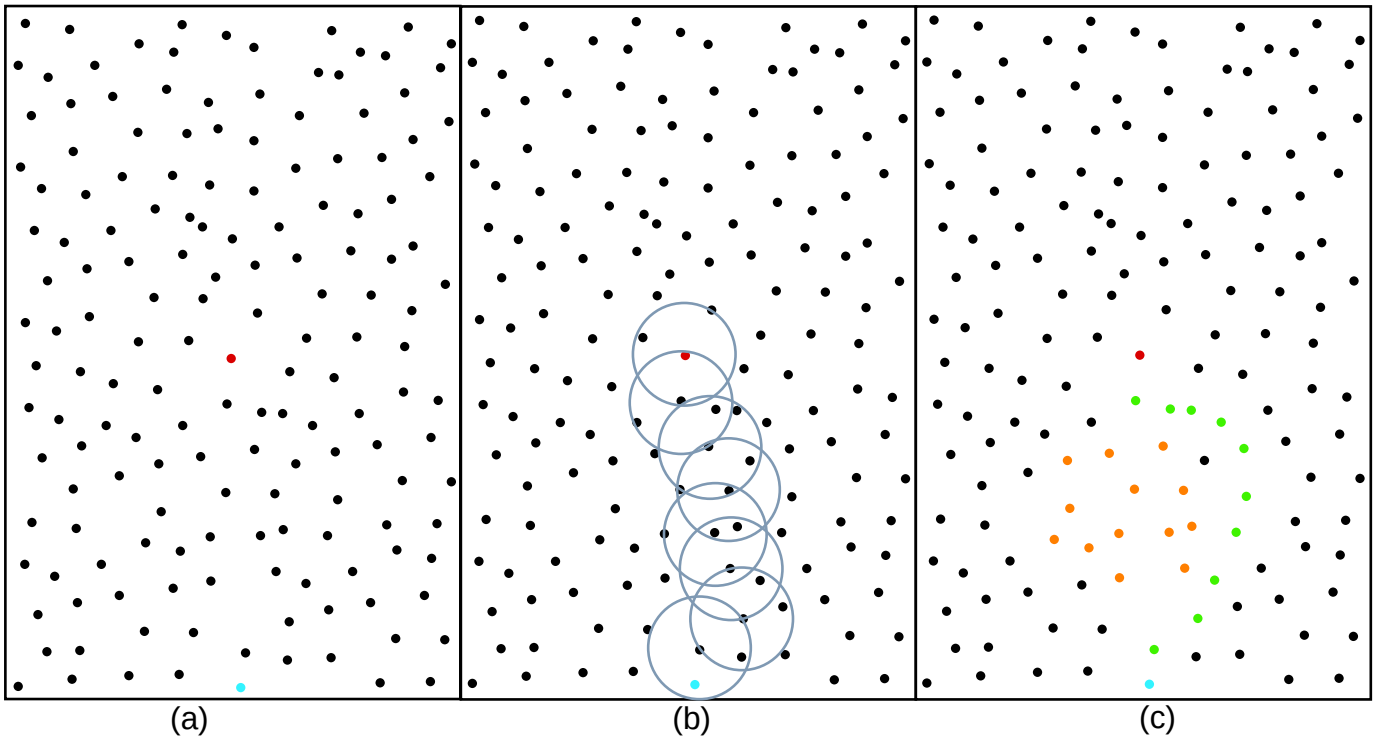

Fig. 2. **Illustration of the simulation environment:** (a) shows a typical simulated graph with randomly placed nodes. The sink is located in the center of the graph with a randomly chosen node many hops away on the bottom of the graph. (b) shows the connectivity of various nodes in a randomly chosen path from the source to the sink. (c) shows a failure introduced into the network between source and sink. To the right of the failure are nodes that may be used to route around the failure.

bandwidth consumption not only causes congestion, collisions, and packet loss, but also requires additional energy. We show in Figure 4 that the additional processing per node is low for high density sensor networks.

Wireless sensor networks provide the unique opportunity to utilize the broadcast medium not only for neighborhood monitoring but also to provide additional resiliency to failures. We tested the effect of passive participation as defined in section I. The probability of participating was varied from zero to six percent. Remember that passive participation only occurs when a node perceives a packet to be lost–it does not necessarily occur at every hop. The results of passive participation shown in Figure 5 are encouraging. For a five percent chance of processing packets perceived as dropped, we attain an additional fifteen percent event delivery ratio, an increase of 28% over the reliability of ARRIVE without passive participation.

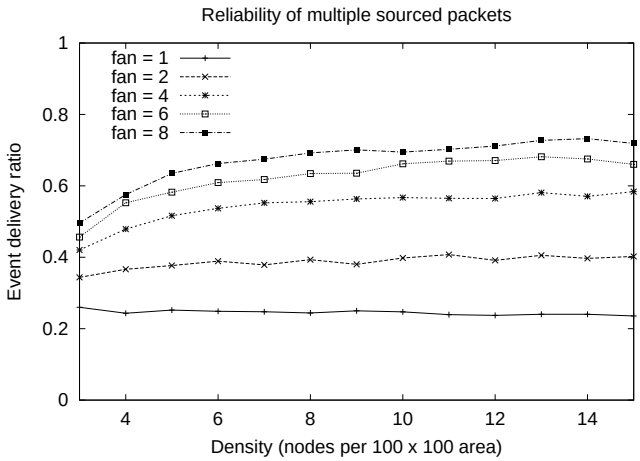

Fig. 3. **Multiple sourced packets:** This illustrates the ratio of events received with respect to density. For one sourced packet, the ratio is extremely low; however the reception dramatically increases with additional sourced packets for each event.

B. Beam-forming

Although packets take a random path from the source to the sink, there needs to be some measure of containment to prevent excessive wandering and routing loops. Forwarding probabilities are increased in proportion to the current level after each push in order to gently "encourage" packets to move towards the sink.

Bandwidth and energy closer to the sink is more valuable than that at higher numbered levels. Nodes at lower levels are required to route all packets traveling to the sink and may lose energy more quickly than other nodes because of this. In addition, since the corresponding low level links are burdened with more traffic, the likelihood

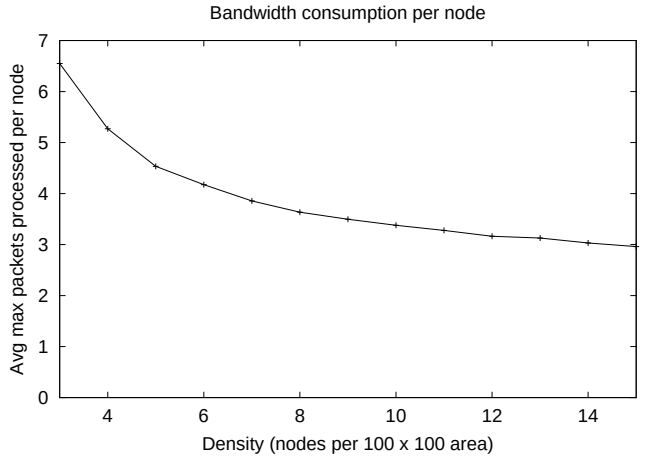

Fig. 4. **Bandwidth:** For eight sourced packets per event, this figure depicts the bandwidth consumption. One event occurs per simulation and the maximum number of packets processed by any given node for the entire simulation is recorded. The results of a series of simulations are averaged.

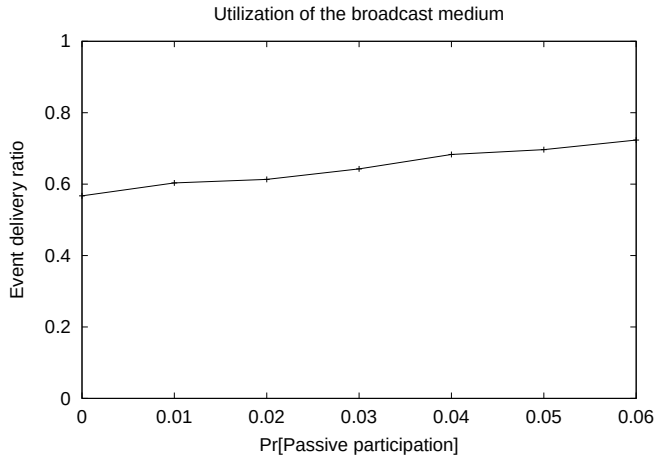

Fig. 5. **Passive participation:** Utilizing the broadcast medium, this figure shows the improvement in event reception over various probabilities of participating in passive packet processing. These results were recorded for four sourced packets on graphs of density equal to ten.

of congestion and collisions is higher than in links farther from the sink. The method in which forwarding probabilities are modified are sensitive to this. Packets that choose to push close to the sink have their forwarding probability increased more dramatically than those that push farther away, firmly suggesting to those packets "almost there" to conclude their journey promptly.

Informally, our goal is to restrict the magnitude of wandering of a packet in proportion to the level of the node that sourced it. This property is expressed in Figure 6. Convergence detection ensures that the convergence of two packets at a node representing the same event are forwarded on different outgoing links. This accounts for the slight increase in the ratio of additional hops to the level of the sourcing node as the fanout increases.

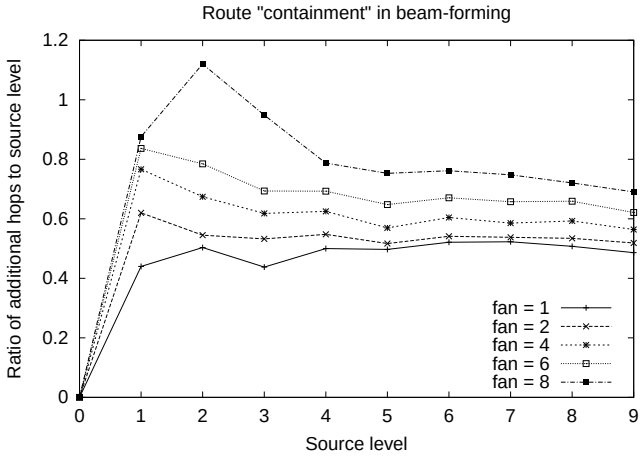

Fig. 6. **Route "containment":** This shows the relative number of additional hops (actual route length - shortest route length) packets take while traveling from a source to the sink for various fanouts. The metric measured is the ratio of the additional hops to the level of the source.

C. Patterned node failures

Figure 7 analyzes ARRIVE's ability to route around patterned node failures. The first observation is that sourcing only a single packet per event performs very badly in the presence of even the smallest failures. But note that even when the fanout is 1, the algorithm is still probabilistic. In the deterministic protocol implemented in TinyOS, even a small failure that is well-placed would cause the event delivery ratio to be 0. Second and more importantly, the magnitude of increased reliability outpaces the corresponding increase in fanout. For failures of size 30, reliability can be improved 8x with a corresponding increase in fanout of only 4x. The magnitude of gains is even greater for larger size failures. Beyond a fanout of 4, the returns are diminished. With a 50% increase in fanout from 4 to 6, there is a corresponding gain of between 10%-50% in reliability (depending on failure size), and there is little to gain by increasing the fanout from 6 to 8.

The reason for the initial extraordinary gains is because of a positive feedback-loop. As we start to increase the fanout, more traffic is injected into the network, more bad nodes are discovered in a shorter period of time, and future traffic is more likely to choose a good path. The effect of this is dramatic as we initially increase the fanout, but eventually becomes overly redundant, and the benefits are diminished.

As expected, the closer the center of failure is to the sink, the more dramatic the effect on reliability. If there are a large number of failures at level one or two, packets

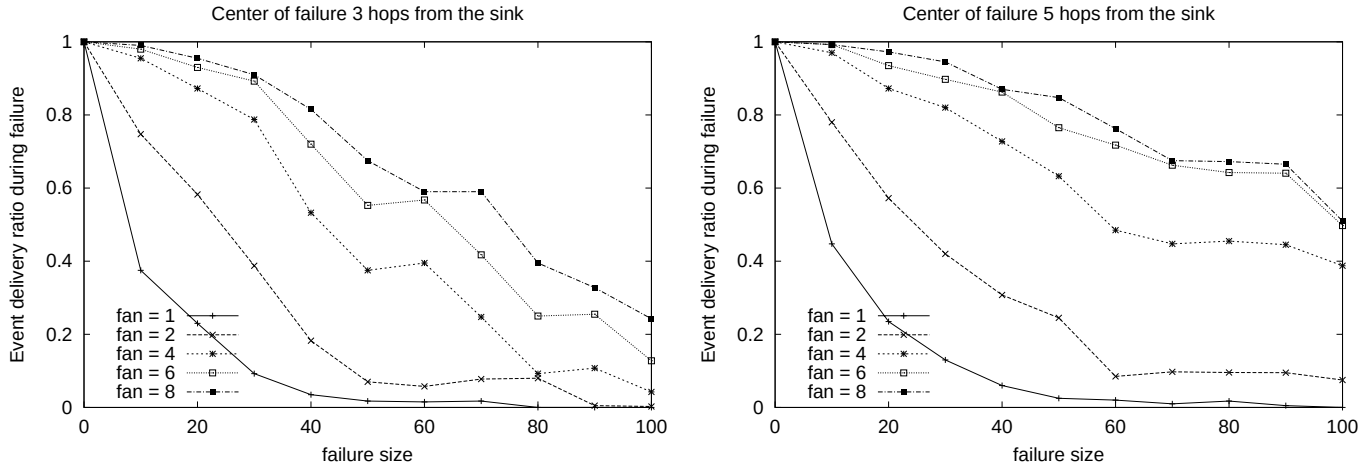

Fig. 7. **Resilience to node failures:** Effect of fanout size on event delivery ratio for various size failures. Failures were tested at two levels of the network: 3 and 5 hops from the sink.

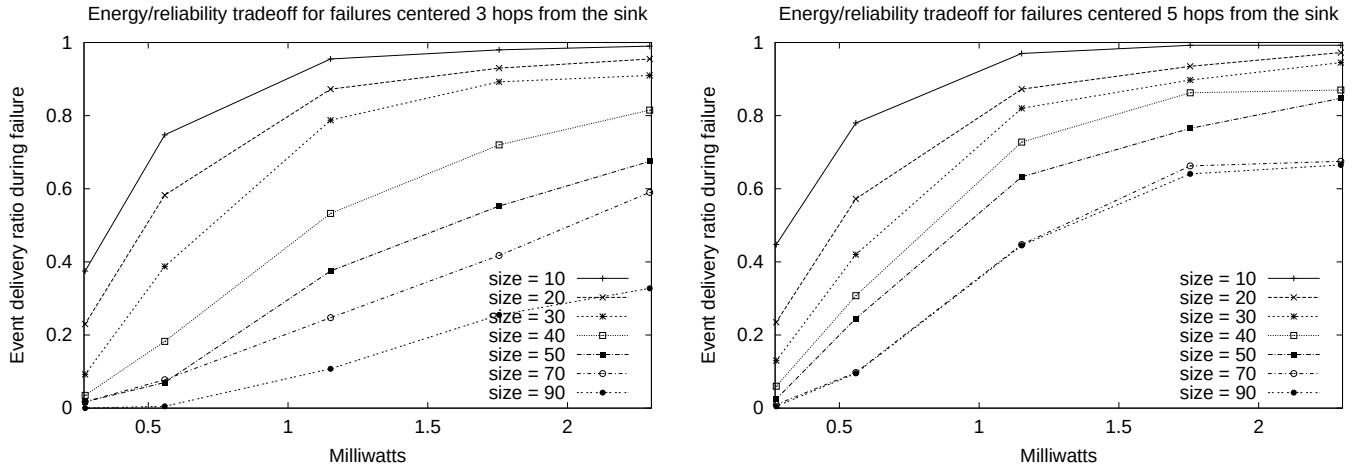

Fig. 8. **Exploring the energy consumption/reliability tradeoff** for various size failures. Energy consumption is the average energy consumption of the network per second during normal(failure free) operation. Reliability is the event delivery ratio during the failure period.

sourced at lower levels must find more circuitous routes to the sink. In addition, forwarding probabilities are updated in a way that discourages alternate path exploration as packets move closer to the sink (see discussion in Section V-B). This strongly motivates future work to develop special countermeasures for failures near the sink.

D. The energy/reliability tradeoff

The reliability gains of sourcing multiple packets are not free. An important issue in the analysis of ARRIVE is the corresponding cost in terms of energy consumption. Figure 8 explores this tradeoff. The graphs is Figure 8 relate the average power consumption per second of each active node[1] during normal operation (no failures) to the event delivery ratio during failures. Although sensor networks are lossy environments, we expect large patterned failures to occur relatively infrequently (although still more frequently than conventional ad-hoc wireless networks). What is the increase in power consumption during normal operation for a corresponding increase in resiliency during times of failure? Intuitively, the proportions should be similar to the conclusions made regarding Figure 7, but it doesn't necessarily follow. In Figure 7 we are comparing the same metric (different levels of reliability during times of failure), but in Figure 8 we are comparing different metrics measured at different times (energy consumption during normal operation vs. resiliency during failure).

It turns out that the magnitude of the gains are similar. By increasing the power consumption by a factor of 4x from its lowest level, we get a corresponding increase

[1]nodes who neither received nor sent a packet during the simulation are excluded

in reliability of 8x. With a further increase of 50%, the returns range from 10%-50% but eventually flatline. A steeper slope indicates better returns on increased power consumption. Similar to the conclusions drawn in Section V-C, the benefits of increased power consumption are greater when failures occur farther from the sink.

E. MICA Platform

We completed an implementation of ARRIVE for the MICA platform. The implementation was written in TinyOS [6], an event-driven operating system for tiny networked sensors. Although the implementation was not debugged by the time of this paper, the implementation is feature complete.

We were able to implement ARRIVE in 10.3 kilobytes of instruction memory out of an available 128 kilobytes. The used instruction memory includes not only the ARRIVE algorithm but also the operating system and its services including the wireless network radio stack. We estimate that the operating system and its services occupy approximately half of the instruction memory, 5.3 kilobytes. Additionally, the memory overhead for detecting convergences, keeping neighbor state, and temporary storage is 737 bytes. The operating system occupies an additional 329 bytes of memory. The memory usage could be significantly reduced by the implementation of a dynamic memory allocator for TinyOS and the use of hash tables.

We have demonstrated that ARRIVE is feasible for sensor networks. Using a small variable and instruction memory footprint, ARRIVE may be used on sensor network nodes in addition to the data acquisition and processing applications which utilize our algorithm for reliable routing.

F. Routing security

Although we did no quantitative security analysis of ARRIVE, we will briefly discuss it here. The notion of security that ARRIVE tries to meet is when the network contains compromised nodes. Efficient cryptographic protocols targeting sensor and ad-hoc wireless networks have been proposed [1], [22], [23], [24], [25], but these methods become ineffective in the presence of compromised nodes. Node compromise is always a threat in any network, but even more so in sensor networks. Physical security of nodes is extremely weak. Sensor nodes are anticipated to be "scattered" in the deployed environment, which makes it easy for resourceful adversaries to obtain a couple of nodes, download any secret keys, inject some malicious code, and re-deploy them. Compromised nodes have all the rights and privileges of legitimate nodes, so they are difficult to defend against. ARRIVE could be augmented with mechanisms from [1] to protect against external adversaries, but its own mechanisms provide a solid foundation for a routing architecture resilient to compromised nodes. Nodes in ARRIVE trust very little information from their neighbors and are less susceptible to bad or stale routing information. The probabilistic nature of ARRIVE enables packets to take disjoint routes and circumvent malicious nodes. It is difficult for a malicious node to draw any more traffic to itself than a friendly well-behaved node.

Passive attacks (refusing to forward packets) are similar to node failures, but active attacks are more difficult to handle. Malicious nodes can lie about their level, replicate themselves to draw more than its share of traffic, and forward packets into oblivion to increase its reputation. We have started to develop heuristics to help counter these attacks, including passive participation. Although ARRIVE is far from being a secure routing protocol, we believe its probabilistic nature and few trust relationships make it a solid foundation to build on.

VI. Conclusion

Our results clearly indicate that routing algorithms for sensor networks should leverage the broadcast medium. Using the medium results in less packets sent, resiliency to failures, a more secure and robust network environment, and great throughput in unreliable networks. Robustness is provided through maintaining neighbor reputations. Not only does it eliminate the sharing of reputations with other nodes which may misrepresent their data, the reputations yield quick recovery and good reliability in the presence of failures of various sizes. ARRIVE's main strength comes from its use of randomness and probabilities to reinforce reliable neighbors and promote packets to travel along independent paths.

Future work for ARRIVE includes further analysis on the Berkeley MICA platform. We intend to study other algorithms for destination selection and probability adjustment. A deeper analysis of the energy tradeoffs and switching to a TDMA MAC layer should result in further power savings.

ARRIVE is a robust routing algorithm for wireless sensor networks. By routing probabilistically over many paths, we prevent any single failure from significantly affecting event data from reaching the sink, or data-collection point. We have demonstrated that ARRIVE is feasible and efficient for routing in sensor networks, especially sensor networks that are volatile and prone to failures.

REFERENCES

[1] A. Perrig et al., "Spins: Security protocols for sensor networks," in *Proceedings of Mobile Networking and Computing 2001*, 2001.

[2] Vincent D. Park and M. Scott Corson, "A highly adaptive distributed routing algorithm for mobile wireless networks," in *IEEE INFOCOM '97*, 1997, pp. 1405–1413.

[3] David B Johnson and David A Maltz, "Dynamic source routing in ad hoc wireless networks," in *Mobile Computing*, Imielinski and Korth, Eds., vol. 353. Kluwer Academic Publishers, 1996.

[4] Charles Perkins and Pravin Bhagwat, "Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers," in *ACM/SIGCOMM'94 Conference on Communications Architectures, Protocols and Applications*, 1994, pp. 234–244.

[5] Chalermek Intanagonwiwat, Ramesh Govindan, and Deborah Estrin, "Directed diffusion: A scalable and robust communication paradigm for sensor networks," in *Proceedings of the Sixth Annual International Conference on Mobile Computing and Networks (MobiCOM '00)*, August 2000.

[6] Jason Hill, Robert Szewczyk, Alec Woo, Seth Hollar, David Culler, and Kristofer Pister, "System architecture directions for networked sensors," in *Proceedings of ACM ASPLOS IX*, November 2000.

[7] Y. Yu, R. Govindan, and D. Estrin, "Geographical and energy aware routing: A recursive data dissemination protocol for wireless sensor networks," 2001.

[8] Brad Karp and H. T. Kung, "GPSR: greedy perimeter stateless routing for wireless networks," in *Mobile Computing and Networking*, 2000, pp. 243–254.

[9] Van Jacobson, "Congestion avoidance and control," in *ACM SIGCOMM '88*, Stanford, CA, Aug. 1988, pp. 314–329.

[10] C. Perkins and E. Royer, "Ad-hoc on-demand distance vector routing," in *MILCOM '97 panel on Ad Hoc Networks*, 1997.

[11] Deepak Ganesan, Ramesh Govindan, Scott Shenker, and Deborah Estrin, "Highly-resilient, energy-efficient multipath routing in wireless sensor networks," *Mobile Computing and Communications Review*, vol. 4, no. 5, October 2001.

[12] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris, "Span: an energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks," *ACM Wireless Networks Journal*, vol. 8, no. 5, September 2002.

[13] Y. Xu, J. Heidemann, and D. Estrin, "Geography-informed energy conservation for ad hoc routing," in *Proceedings of the Seventh Annual ACM/IEEE International Conference on Mobile Computing and Networking*, 2001.

[14] Frank Stajano and Ross J. Anderson, "The resurrecting duckling: Security issues for ad-hoc wireless networks," in *Seventh International Security Protocols Workshop*, 1999, pp. 172–194.

[15] Inonut Aron and Sandeep Gupta, "A witness-aided routing protocol for mobile ad-hoc networks with unidirectional links," 2000.

[16] Sergio Marti, T. J. Giuli, Kevin Lai, and Mary Baker, "Mitigating routing misbehavior in mobile ad hoc networks," in *Mobile Computing and Networking 2000*, 2000, pp. 255–265.

[17] A. Demers, S. Shenker, V. Bhargavan, and L. Zhang, "Macaw: A media access protocol for wireless lans," in *ACM SigComm '94*, 1994.

[18] Jason Hill and David Culler, "A wireless embedded sensor architecture for system-level optimization," in *Submission to USENIX ASPLOS '02*, 2002.

[19] D. Ganesan, B. Krishnamachari, Alec Woo, David Culler, Deborah Estrin, and Stephen Wicker, "An empirical study of epidemic algorithms in large scale multihop," in *Submission to the Eighth Annual International Conference on Mobile Computing and Networks (MobiCOM '02)*, March 2002.

[20] Lee Breslau, Deborah Estrin, Kevin Fall, Sally Floyd, John Heidemann, Ahmed Helmy, Polly Huang, Steven McCanne, Kannan Varadhan, Ya Xu, and Haobo Yu, "Advances in network simulation," *IEEE Computer*, vol. 5, no. 33, May 2000.

[21] M. Stemm and R.H. Katz, "Measuring and reducing energy consumption of network interfaces in hand-held devices," *IEICE Transactions on Communications*, vol. E80-B, no. 8, August 1997.

[22] L. Zhou and Z. Haas, "Securing ad hoc networks," *IEEE Network Magazine*, vol. 13, no. 6, November/December 1999.

[23] Weilin Zhong and David Evans, "When ants attack: Security issues for stigmergic systems," 2002.

[24] J. Hubaux, L. Buttyan, and S. Capkun, "The quest for security in mobile ad hoc networks," in *Proceedings of the ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHOC 2001)*, 2001.

[25] J. Kong, P. Zerfos, H. Luo, S. Lu, and L. Zhang, "Providing robust and ubiquitous security support for mobile ad-hoc networks," in *ICNP*, 2001, pp. 251–260.