TinyTermite: A Secure Routing Algorithm on Intel Mote 2 Sensor Network Platform

Mina Sartipi and **Josh Patterson**
Department of Computer Science and Engineering
University of Tennessee at Chattanooga
Chattanooga, TN 37403-2598
Email: {Mina-Sartipi, Josh-Patterson}@utc.edu

Abstract

In this paper, we introduce *TinyTermite*. TinyTermite is a novel AI routing algorithm that is reliable and secure against selective forwarding and replay attacks. TinyTermite is fully implemented on TinyOS based Intel Mote 2 platform and the experiments were done to compare its performance with that of the traditional Termite algorithm. Our results show that the TinyTermite reduces the energy consumption by more than 30% and enhances the security of the network. The experimental results show that TinyTermite is significantly more secure against replay and sinkhole attacks by lowering the packet loss from 88.5% to 32.9% with 12.7% normal packet loss. Finally, the experimental results also demonstrates that the TinyTermite provides high throughput and low latency.

Introduction

In wireless sensor networks, packets and queries are sent between source and sensor motes, or from one mote to another. The routing algorithm finds the most efficient path(s) for this transmission. In this paper, we consider a dynamic wireless sensor network, where sensor motes can be added to, removed from, or moved around the network. Sensor motes have limited battery capacity and they must work for a satisfactory period of time and cannot be charged during operation. Energy is consumed by motes in their sensing, processing and communication tasks. Processing and communication energy consumption depends not only on the hardware, but also on the way data is communicated. Therefore, energy is one of the major constraints on communication schemes. In addition to the importance of energy efficiency and longevity of a dynamic wireless network, security is another important measure of performance for the system. In this work, selective forwarding and replay attacks are investigated. Moreover, these networks are deployed with very little backbone infrastructure and assuming knowledge of network topology is not realistic in practice. Thus, the routing algorithm needs to be adaptable, efficient, and simple.

We propose an AI multihop routing algorithm that is suitable for ad-hoc wireless sensor networks considering the above constraints. We secured existing Termite algorithm (Roth & Wicker 2005) against selective forwarding and replay attacks. This new algorithm is called TinyTermite. We introduced a suspicion technique that is inspired by task response thresholds in ant colonies. This technique detects certain classes of attacks even given inconclusive information. We implemented the proposed TinyTermite on Intel Mote 2 wireless sensor network platform (Adler *et al.* 2005) and measured the systems's performance in a real-world setting.

Previous Works

In general, routing in wireless sensor networks can be divided into flat-based routing (SPIN (Kulik, Heinzelman, & Balakrishnan 2002), Directed Diffusion (Intanagonwiwat, Govindan, & Estrin 2000), Rumor Routing (Braginsky & Estrin 2002)), hierarchical-based routing (LEACH (Heinzelman, Chandrakasan, & Balakrishnan 2000), PEGASIS (Lindsey & Raghavendra 2002), TTDD (Ye *et al.* 2002)), and location-based routing (GEAR (Maymounkov 2001), SPAN (Chen *et al.* 2002)). Among these algorithms, only SPIN and TTDD can be applied to mobile WSN (Al-Karaki & Kamal 2004). SPIN is suitable for environments where the sensors are mobile, but SPIN's data advertisement mechanism cannot guarantee the delivery of data (Al-Karaki & Kamal 2004). In TTDD, sinks may change their locations dynamically, whereas sensor nodes are stationary (Al-Karaki & Kamal 2004). In the dynamic networks, both sensor and sink may change their location dynamically. Therefore, we investigate the application of ad-hoc network routing algorithms for WSN. The application of ad-hoc routing algorithms to WSNs has also been studied in (Camilo, Silva, & Boavida 2006; Shin, Ramachandran, & Ammar 2007).

The routing algorithms in mobile ad-hoc networks are generally based on distance vector or link state (Bellman 1958; Jr. & Fulkerson 1956; Dijkstra 1959; Perkins & Bhagwat 1994; Jacquet *et al.* 2001; Perkins & Royer 1999) and fall into two categories, proactive and reactive. OLSR (Jacquet *et al.* 2001) and DSDV (Perkins & Bhagwat 1994) are examples of proactive techniques, while DSR (Johnson & Maltz 1996) and AODV (Perkins & Royer 1999) are considered to be reactive. Detailed simulation comparative studies of proactive and reactive protocols have shown that AODV and DSR outperform DSDV and OLSR at all mobility speeds (Broch *et al.* 1998; Clausen, Jacket, & Viennot 2002) and AODV outperforms DSR in stressful scenarios, where stressful networks are considered to be dense and highly mobile (Tomar 2008). Al-

though AODV outperforms the other routing algorithms, under highly dynamic environments its performance degrades significantly.

In (Roth & Wicker 2005), the authors show that Termite, a biologically-inspired routing algorithm, outperforms AODV due to the lack of control traffic and effectively liberal route caching. Therefore, our proposed routing algorithm suited for dynamic environment will be based on the approaches used by Termite algorithm. In the next section, Termite is briefly reviewed.

Background Review- Termite

Termite is a probabilistic algorithm in which routing decisions for packets are primarily influenced by a routing metric, termed *pheromone*. Pheromone is analogous to the chemical pheromone laid down along the path of an individual ant which is used by other members of ant colony to determine their likely direction of travel. Similarly, Termite relies on digital pheromone levels to probabilistically choose the next hop for a forwarded packet. The higher the pheromone level associated with a link and destination, the more likely it will be chosen as the next hop for a packet with the same destination. As packets traverse the network they carry a pheromone value which is used to *deposit* pheromone along the links on which they travel. This deposition is associated with the link along which the packet has come from, rather than on the link it will be forwarded on. This method of updating the pheromone table results a local pheromone map in a direction opposite that of network traffic. As pheromone is deposited, the pheromone value stored in the packet is proportionally decreased.

As with most reactive systems, an effective implementation requires a balance between positive and negative feedback. In an ant colony, positive feedback is accomplished by the continual deposition of pheromone as individual ants travel. However, if negative feedback was not introduced, the area around the colony would become saturated with pheromone and it would afford no directive benefit to the colony. In this case, evaporation of the pheromone molecules provides the necessary negative feedback, allowing the density of the deposited pheromone to shift location over time as ants abandon a particular pathway. In Termite, a virtual *evaporation* function provides negative feedback that causes the pheromone associated a given link to decay towards zero over time. Thus, if a neighboring node becomes nonfunctional for some reason, the pheromone associated with its link will decrease until the local node will no longer attempt to forward packets along it. In essence, Termite's internal routing tables are tuned based on packet movement as observed locally rather than relying on explicit routing information received from other neighbor nodes. This generally results in much lower overhead compared with protocols which must rely on the exchange of control packets to update their routing tables.

In addition to adaptability, the routing algorithm must be secure. The communication in wireless sensor networks can be disrupted in a number of malicious ways such as selective forwarding and replay attacks (Karlof & Wagner 2003). Although the nature of Termite algorithm minimizes the effect of these attacks, it cannot prevent them. One possible solution is applying link layer security mechanisms to Termite routing algorithm. These techniques can help mediate some of the resulting vulnerabilities, but cannot eliminate them. In other words, the routing algorithm must be designed with the security in mind. In the next section, TinyTermite is introduced.

TinyTermite

We propose TinyTermite as a routing algorithm that is secure in a malicious environment and robust against node failure, movement, and expendability. To achieve these, TinyTermite has a reactive probabilistic design, similar to Termite, intended to maximize throughput, but unlike Termite, it is designed with security in mind. To provide robustness against noisy environments and distance fading, rateless coding (Pakzad, Fragouli, & Shokrollahi 2005) is applied on each packet before sending. TinyTermite also has a suspicion pheromone table that secures the network against replay and selective forwarding attacks. The suspicious pheromone is increased when the same packet is received more than once. In summary, TinyTermite is effective in a highly dynamic network, robust against failure and node movement, and node expendability.

Securing TinyTermite against replay attacks using suspicion pheromone

Replay Attack In a replay attack, an attacker node will capture or create a valid packet and repeatedly send this packet to a neighbor. To the neighbor, the packet will have valid fields and/or cryptographic information. The neighbor node needs to use resources to process each incoming replayed packet. In a setup where the node is running off limited battery power, this can quickly deplete resources and effectively disable the neighbor node.

With Termite, incoming packets influence a node's pheromone table. Each outbound packet for this node is then influenced by the same pheromone table. If a replayed packet continuously comes from the attacker node, it will heavily influence the pheromone table of the neighbor node. The neighbor node will quickly rate the attacker node as a good next hop, which sets the attacker up for more nefarious behavior as it now sits on a high quality route for the network towards a particular destination. Once this situation is setup, the attacker can then execute a selective forwarding attack where it only forwards a subset of the received packets or none of them at all. The replay attack is demonstrated in Figure 1. To counter this situation and create a basis to counter other attacks, we introduced the technique of Suspicion.

Suspicion Pheromone In mobile ad-hoc networks, it is often difficult for a node to determine if the activity of its neighbor is good or bad (Karlof, Sastry, & Wagner 2004). A technique is required that detects various types of behavior, records events efficiently, and reacts appropriately when needed. As stated above, in Termite each neighbor link simply exists or not exist, and the strength of the connection is

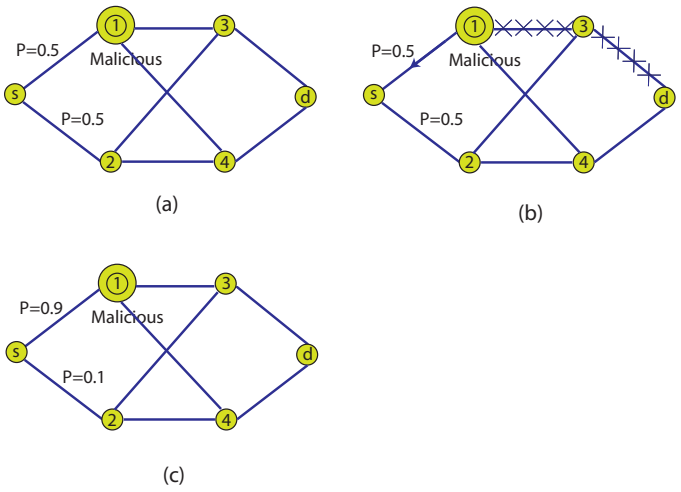

Figure 1: Replay attack on Termite routing algorithm. (a) Source s has packets to send to the destination d. Initially, both neighbors (nodes 1 and 2) have the same pheromone values. (b) The malicious node 1 replays packets and falsely claims the source of the packets are d. (c) The pheromone value on link $(s, 1)$ toward destination d rises.

based on the amount of pheromone on the link. TinyTermite creates a more sophisticated neighbor link relationship as each neighbor is given a level of *suspicion*. This suspicion factor is similar to the pheromone used for each destination, but it represents a small amount of temporally affected information about how much the neighbor is trusted. The higher the suspicion pheromone of a link, the less unlikely that will be chosen as the next hop. Certain types of neighbor behavior are defined that will increase the suspicion pheromone for that link. If this behavior continues and the suspicion pheromone surpasses a threshold level (τ), the node ceases communication with the neighbor node and reports the node as a malicious node. However, if the behavior was simply a result of normal but fluctuating traffic effects, then the degradation of the suspicion pheromone over time eases the penalty on the offending node.

The suspicion pheromone will be updated and degraded according to the following equations, respectively:

$$S_{i,j} \leftarrow S_{i,j} + \delta, \tag{1}$$

$$S_{i,j} \leftarrow -\lfloor \frac{t}{t_o} \rfloor d + S_{i,j}, \tag{2}$$

where $S_{i,j}$ denotes the level of suspicion pheromone on the edge (i, j) linking i to its neighbor j, δ represents the level of suspicion pheromone deposited on the links, and d is the constant amount by which the suspicion pheromone degrades. At each time interval (t_o), the pheromone degrade equation is applied to each existing neighbor link in the routing table.

To secure TinyTermite against replay attacks, the algorithm is designed so that suspicion is raised when the node *sees* a packet that it had received before. To implement the detection scheme for this stimulus, a *seen-packet* list is implemented. This list keeps the ID of the packets it has seen in the last N seconds, where N is the route timeout factor used in routing the packets. When node i receives a packet from node j with a packet ID that exists in its *seen-packet* list, the suspicious pheromone for link (i, j) builds up according to the above equation. However, if the behavior was simply a result of normal but fluctuating traffic effects, then the degradation of the suspicion pheromone over time eases the penalty on the offending node.

The process of securing TinyTermite using suspicion pheromone is explained in Fig. 2. The pheromone table of a node has two values of pheromone and suspicion pheromone for each outgoing link. These values are denoted by $[P, SP]$. In this figure, source s has packets to send to the destination node d. At the beginning node s has the same amount of pheromone (0.5) and suspicion pheromone (0) for links $(s, 1)$ and $(s, 2)$. The attacker node 1, replays packets whose sources are falsely claimed to be d. This will increase the pheromone value for link $(2, 1)$. As this replay attack continues, the suspicion pheromone starts rising. Once the suspicion amount for the link (i, j) rises past the associated threshold (τ), the link is now in timeout status. The source node will not process the packets coming from 1, nor send any packet to it until its suspicion pheromone level decays below the threshold. This link is back into service as soon as its suspicion level drops below the threshold. However, the suspicion level is now near the threshold and the link is still on the verge of being put right back in timeout given the slightest provocation. This creates a flexible sort of memory for the network; it slows down and isolates regions that are misbehaving while giving them a chance to come back into normal operation over time.

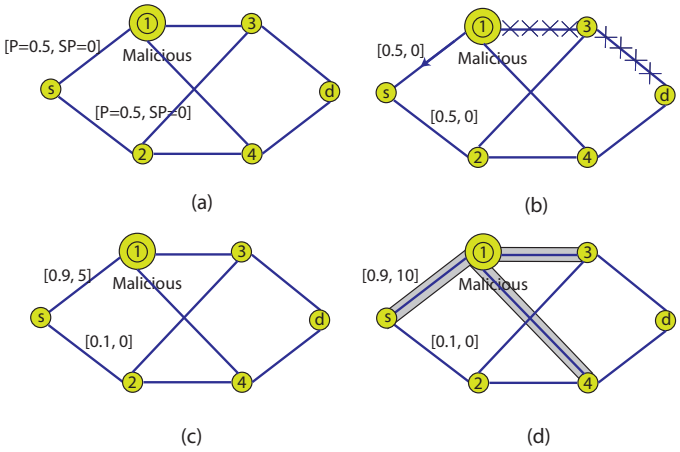

Figure 2: Replay attack on TinyTermite routing algorithm. (a) Source s has packets to send to the destination d, both neighbors (nodes 1 and 2) have the same pheromone values. (b) The malicious node 1 replays packets and falsely claims the source of the packets are d. (c) The pheromone value on link $(s, 1)$ toward destination d rises and so does the suspicion pheromone. (d) The suspicion pheromone value has passed the threshold value. Mote 1 is a suspicious node now and all the links ending at this node are temporarily unavailable (the shaded links).

Analysis on Suspicion Pheromone For the experiments, $\delta = 3$, $d = 0.5$, $N = 4000$ms, and threshold for suspicion of $\tau \in [7.5, 10.5]$ were used. Considering the practical throughput for Intel Mote 2 (reported in the last Section) and the packet size, a total of 9KB of space is used to track messages coming through the node, which is feasible for Intel Mote 2.

When the threshold is known, a different type of attack pattern can be applied to the network. The attacker replays the same packet until the threshold is near and stops to allow the pheromone to go down. In TinyTermite the exact value of threshold is not known. Each node has a random threshold value that is uniformly chosen from $[\tau_{\min}, \tau_{\max}]$. TinyTermite cannot prevent these attacks, but it significantly limits their effectiveness, since the attacker cannot flood the network and it has to stop the attack between attacking intervals. The ongoing research is investigating ANN pattern recognitions to determine the suspicious pheromone. This method, will observe suspicious acts over a time interval rather than at any time stamp. At the current stage, the cyclic pattern described above is being studied. Investigating other patterns is part of the future work.

Experimental Results

Our experiment set consists of 20 sensor motes randomly distributed. The power setting is chosen such that communication range is about 1 foot, i.e., radio power setting of less than 3. The motes use TinyTermite to transmit information. To evaluate the routing scheme, we implemented the TinyTermite and experimentally measured its performance costs. In these experiments, we empirically determined TinyTermite's impact on total number of transmissions (we consider the energy spent for RF transmission as in (Wieselthier, Nguyen, & Ephremides 2002)), security, throughput, and latency. TinyTermite is also compared with Termite with respect to the above parameters. Since TinyTermite behaves similar to Termite in dynamic networks and Termite's effectiveness in dynamic networks is shown, we provide our results on static networks.

Energy Consumption

In these experiments, we consider the energy spent for RF transmission. The goal was to send 1000 packets from the source mote to the destination mote using two methods. Both Termite and TinyTermite were implemented to find the energy consumption. In both methods, the relaying motes, upon forwarding any packet, update the total number of packets they have forwarded. In Termite, the receiver keeps track of the packets it has not received and asks for retransmission. This process is continued till all 1000 packets are received. Then, the total number of packets are multiplied by r^2, where r is the distance between motes to find the total transmission energy. In the TinyTermite method, rateless coding is applied on the original packets and the generated encoded packets are forwarded. The source continues generating encoded packets and forwarding them, till it hears the ACK from the destination mote. This ACK confirms that sufficient packets were received and decoding was done successfully at the destination mote. Similar to the first method, the total number of packet transmissions are multiplied by r^2. The results revealed that the total number of transmissions for sending of 1000 packet over the mentioned setup is equal to 1698 and 1263, for the first and TinyTermite method respectively. As we can see, TinyTermite reduces the energy consumption by more than 30%.

Table 1: Total number of transmissions for sending 1000 packets from WMS to the destination mote over the experimental network.

Method	Total Number of Transmissions
Termite	1698
TinyTermite	1263

Security

For the suspicion test we used the graph shown in Fig. 2(a), where source sends packets to the destination mote. The malicious node simulate replay and sinkhole attacks. The replay attack repeatedly sends out a captured valid packet at high rates which can degrade and drain resources in the network. The malicious node influences the Termite algorithm by influencing the pheromone table to give more weight to the neighbor with replay attack. The replay attack makes the network to bias routing via malicious node setting up the sinkhole attack. A sinkhole attack is when a node simply does not forward packets, which in this case provides an effective attack. Our malicious node was setup to send out a replayed packet every second, and the source node queued five normal packets to be routed to the destination node every second. The malicious node and the source mote both began sending packets at the same time. For each session the source queued five packets a time until reaching 100 total packets. We ran each set of experiments 50 times for the cases of no malicious node, malicious node attack without suspicion defense (Termite algorithm), and malicious node attack with suspicion defense (TinyTermite algorithm). The results of the experiments are shown in Table 2. It is worth noting that in these experiments, rateless coding was not applied.

The baseline packet loss for this setup was measured to be 12.7% packet loss. With the malicious node actively attacking the network, the packet loss reached an average of 88.5% and severely disrupted the network communication. Now with the malicious node still actively attacking the network but activated the suspicion defense mechanism, the packet loss dropped to 32.9%. Considering that our normal drop rate is 12.7% for the same network without the malicious node, the suspicion defense was able to significantly lower the packet loss in the network under attack.

Throughput

To measure the throughput of TinyTermite, a source node sends out a sequence of packets to a destination node. The destination node, upon receiving the first packet, records the

Table 2: Comparing the security of TinyTermite algorithm with Termite algorithm. The experiments were done on network of Fig. 2(a) whose baseline packet loss was measure to be 12.7%.

Method	Number of Packets Received by the Destination Mote	Number of Packets Captured by the Malicious Node	Packet Loss Rate
Termite	11.5	70.2	88.5%
TinyTermite	67.1	12.9	32.9%

time it was received and increments a packet counter. Subsequent packets also increment the packet counter, but update a separate time record which tracks the reception time of the most recently received packet. Upon completion of the test, the destination node calculates the throughput by dividing the number of packets received multiplied by the size of the data payload, divided by the difference between the time the last packet was received and the first packet was received. This provides a value for the throughput in bytes per second.

To determine the impact of the TinyTermite on throughput, first the throughput between two isolated Intel Mote 2s was measured to be 2.9 KB/s (raw throughput). Next, a line of motes was set up at 1 foot intervals so that each node had two neighbors. The motes were pre-programmed with static routes which forced the packets to take a predetermined route. This effectively created a line of motes with the source node at one end and the destination node at the far end. The throughput of this configuration was measured to be 2 KB/s. Finally, the TinyTermite was implemented on our experiment set up and the throughput was measured to be 1.84 KB/s. The experimental measurements show that the increased packet size has little impact on the throughput. Table 3 summarizes the results.

Table 3: Effect of TinyTermite on throughput

Raw throughput	2.9 KB/s
Throughput of line network with pre-determined paths	2KB/s
TinyTermite throughput	1.84 KB/s

Latency

Measurement of the the latency of the TinyTermite implementation was accomplished by the transmission of a packet from a source node to a destination node. Upon reception of this packet, the destination node sends back the packet toward the source. Immediately upon sending the initial packet, the source node records time, then when it receives the return packet, it again records the time. The difference between these two times provides the round-trip-time of the communication. Latency on our experimental setup graph was 47.81 ms on average based on 50 runs of 1000 packets for each run.

Conclusion

We have presented TinyTermite, our novel probabilistic routing algorithm. TinyTermite is fully implemented on TinyOS based Intel Mote 2 nodes. Our proposed routing algorithm provides adaptability to dynamic network topology, security, scalability, and robustness. Our experimental results have demonstrated the significant security enhancement as well as energy consumption of TinyTermite versus traditional Termite algorithm while still keeping the throughput high and latency low.

References

Adler, R.; Flanigan, M.; Huang, J.; Kling, R.; Kushalnagar, N.; Nachman, L.; Wan, C.; and Yarvis, M. 2005. Intel Mote 2: An advanced platform for demanding sensor network applications. *Proc. ACM Embedded Networked Sensor Systems* 298 – 298.

Al-Karaki, J., and Kamal, A. 2004. Routing techniques in wireless sensor networks: a survey. *IEEE Wireless Communications* 11(6):6–28.

Bellman, R. 1958. On a routing problem. *Quarterly of Applied Mathematics* 16:87–90.

Braginsky, D., and Estrin, D. 2002. Rumor routing algorithm for sensor networks. *Proc. the First Workshop on Sensor Networks and Applications*.

Broch, J.; Maltz, D.; Johnson, D.; Hu, Y.; ; and Jetcheva, J. 1998. A performance comparison of multihop wireless ad hoc network routing protocols. *Proc. MOBICOM* 85–97.

Camilo, T.; Silva, J.; and Boavida, F. 2006. Assessing the use of ad-hoc routing protocols in mobile wireless sensor networks. *Proc. CSMU*.

Chen, B.; Jamieson, K.; Balakrishnan, H.; and Morris, R. 2002. SPAN: an energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks. *Wireless Networks* 8(5):481–494.

Clausen, T.; Jacket, P.; and Viennot, L. 2002. Comparative study of routing protocols for mobile ad hoc networks. *Proc. First Annual Mediterranean Ad Hoc Networking Workshop*.

Dijkstra, E. 1959. A note on two problems in connection with graphs. *Numerische Mathematik* 1:269–271.

Heinzelman, W.; Chandrakasan, A.; and Balakrishnan, H. 2000. Energy-efficient communication protocol for wireless microsensor networks. *Proc. the 33rd Hawaii International Conference on System Sciences*.

Intanagonwiwat, C.; Govindan, R.; and Estrin, D. 2000. Directed diffusion: a scalable and robust communication paradigm for sensor networks. *Proc. ACM MobiCom* 56–67.

Jacquet, P.; Muhlenthaler, P.; Clausen, T.; Laouiti, A.; Qayyum, A.; and Viennot, L. 2001. Optimized link state routing protocol for ad hoc networks. *Proc. IEEE International Multi Topic Conference* 62–68.

Johnson, D., and Maltz, D. 1996. Dynamic source routing in ad-hoc wireless networks. *Proc. ACM SIGCOMM* 318329.

Jr., L. F., and Fulkerson, D. 1956. Maximal flow through a network. *Canadian Journal of Mathematics* 8:399–404.

Karlof, C., and Wagner, D. 2003. Secure routing in wireless sensor networks: attacks and countermeasures. *Elsevier's AdHoc Networks* 1:293–315.

Karlof, C.; Sastry, N.; and Wagner, D. 2004. TinySec: A link layer security architecture for wireless sensor networks. *Proc. SensSys* 162–175.

Kulik, J.; Heinzelman, W. R.; and Balakrishnan, H. 2002. Negotiation-based protocols for disseminating information in wireless sensor networks. *Wireless Networks* 8:169–185.

Lindsey, S., and Raghavendra, C. 2002. PEGASIS: Power-efficient gathering in sensor information systems. *Proc. IEEE Aerospace Conference* 112–1130.

Maymounkov, P. 2001. Geographical and energy-aware routing: A recursive data dissemination protocol for wireless sensor networks. UCLA Computer Science Department Technical Report, UCLA-CSD TR-01-0023.

Pakzad, P.; Fragouli, C.; and Shokrollahi, A. 2005. Coding schemes for line networks. *Proc. IEEE International Symposium on Information Theory* 1853 – 1857.

Perkins, C., and Bhagwat, P. 1994. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. *Computer Communication review* 234–244.

Perkins, C., and Royer, E. 1999. Ad-hoc on-demand distance vector. *Proc. Mobile Computing Systems and Applications* 90–100.

Roth, M., and Wicker, S. 2005. TERMITE: A swarm intelligence routing algorithm for mobile wireless ad-hoc networks. *Springer SCI Series: Swarm Intelligence and Data Mining*.

Shin, J.; Ramachandran, U.; and Ammar, M. 2007. On improving the reliability of packet delivery in dense wireless sensor networks. *Proc. IEEE Conference on Computer Communications and Networks* 718–723.

Tomar, G. 2008. Modified routing algorithm for AODV in constrained conditions. *Proc. Second Asia International Conference on Modeling and Simulation* 219–224.

Wieselthier, E.; Nguyen, G. D.; and Ephremides, A. 2002. Energy-efficient broadcast and multicast trees in wireless networks. *ACM/Kluwer Journal on Mobile Networks and Applications* 481–492.

Ye, F.; Luo, H.; Cheng, J.; Lu, S.; and Zhang, L. 2002. A two-tier data dissemination model for large-scale wireless sensor networks. *Proc. ACM/IEEE MOBICOM*.