Using History to Improve Lightweight Routing Topology Maintenance in Sensor Networks *

Regi Oommen and Mikhail Nesterenko
Computer Science Department
Kent State University
Kent, OH, 44242
roommen@kent.edu, mikhail@cs.kent.edu

Abstract

An ad hoc wireless sensor network requires a routing topology to efficiently convey the gathered data to exfiltration points. The topology construction and maintenance algorithm for such a network should generate good quality results and efficiently utilize the limited resources of the network. Such algorithm should be agile in its reaction to the environmental changes. Yet it should be stable so that the noise introduced by individual measurements does not result in unnecessary topology modifications. Topology maintenance by flooding is extremely agile and relatively cheap to run. However, the stability and quality of the routing tree topology constructed in a wireless sensor network by flooding is rather poor. In this paper we use experimental data to simulate the dynamics of flooding in a network of Berkeley's prototype sensors. We then study how to improve the stability and quality of resultant routing trees by using the history of message propagation gathered by each individual node. Each node collects the data about previous floods and uses it to optimize parent selection. Specifically, the nodes use either transmission rate or path length as their optimization criterion.

*This research was supported in part by DARPA contract OSU-RF#F33615-01-C-1901.

1 Introduction

Routing maintenance algorithms. A traditional choice for routing topology formation in ad hoc multihop wireless networks are the protocols based either on link-state or distributed distance-vector concepts (cf. DSR [6] and AODV [11]). In case of link-state, the source node has to learn a portion of the topology of the whole network so that all routing decisions can be made at the packet's origin. Even though this centralized decision making eliminates a number of problems that more distributed algorithms face, the communication requirements for collection of routing information and memory requirements for its storage make the link-state based routing algorithms appropriate but for small-scale sensor networks. Distance-vector based algorithms demand significantly less resources. However, these algorithms suffer from a number of inherent stabilization problems such as counting-to-infinity.

Flooding presents a radically different approach to topology construction. In flooding, the destination floods the network with routing messages. Each node selects the sender of the first routing message that it receives to be its parent and propagates the message exactly once. Thus, a routing tree is formed. When the base station needs to refresh the routing tree, a new flood is initiated and a new tree is constructed. Each node stores little information about the rest of the network and discards the old data when new data arrives with the next flood. This makes flooding ex-

ceptionally agile. It requires little memory resources at the nodes and its message complexity is comparable to that of distance-vector and link-state algorithms. However, the haphazard nature of the routing topology construction in flooding yields routes of rather poor quality and these routes tend to completely change with each flood.

Routing maintenance in sensor networks. The platform of choice for this study is Berkeley's prototype sensors [4, 5]. Nesterenko and Arora [9] used the sensors to compare the performance of flooding topology maintenance to a distance-vector based approach. The flooding compared quite favorably both in efficiency and fault tolerance aspects. In a recent experiment, Ganesan et al [2] studied the quality of routing trees constructed by flooding. This study defined the quality metrics for a routing tree constructed by flooding in a wireless environment and identified specific problems associated with such trees.

Study objectives and contribution. The motivation for the present study is to find effective means of improving the stability and quality of the routing topology resulting from flooding. The improvements should not hamper the advantages of flooding (i.e. responsiveness to topological changes and modest resource requirements).

Our approach is for each node to accumulate the information about previous floods and use it to aid in routing tree formation. We selected rather straightforward algorithms of accumulation so that we can directly observe how history influences the routing maintenance characteristics. In the conclusion of the paper we discuss how more involved algorithms can be used to further improve the desired qualities of routing maintenance.

The specific contributions of this paper are as follows. We recreate the experiment of Ganesan et al [2] by running the networked sensors' software in a simulator. We verify the fidelity of our simulation against the experimental data reported in the original paper and in other experimental studies [1] involving Berkeley's prototype sensors. We consider two approaches to utilize data from previous floods. One based on message receipts, the other – on average hop count to the base station. In both cases a moving average with a threshold is used for parent selection. We tune the parameters of both algorithms to achieve optimal adaptive characteristics. Specifically, we observe the performance of the parent selection schemes by varying the threshold size at each selection and by varying the number of floods over which the average is calculated. A small threshold size may lead to quick convergence time but does not improve stability. Likewise, a long record of previous floods may improve stability at the expense of greater convergence time. We measure the quality of the routing trees constructed by these algorithms and compare it against the quality of the trees of the original experiment. We discuss the applications and extensions of history-enhanced flooding in sensor networks.

Related work. In this study we would like to specifically focus on the dynamics and quality of the routing tree construction by flooding in sensor networks. To the best of our knowledge ours is the first study with this objective. However, there is a number of extensively explored adjourning topics. A large number of routing algorithms in general ad hoc networks, such as DSR [6] and AODV [11] use some form of flooding for route discovery, LAR [8] improves on DSR by leveraging geographic information to limit the flood propagation. GOSSIP [3] optimizes communication overhead by probabilistically scheduling route broadcasts. Ni et al [10] study the "broadcast storm" problem of flooding. Kim and Noble [7] discuss a variety of sophisticated techniques to estimate link reliability in a mobile wireless network.

The rest of the paper is organized as follows. We give the details of the original flooding algorithm and the two history-based algorithms in Section 2. We describe the experimental platform, the setup of the experiments and the metrics of evaluating the quality of the routing trees in Section 3. In Section 4 we provide and analyze the results of the experiments. We conclude the paper by discussing the applications and further extensions of this research in Section 5.

2 Topology Maintenance Algorithms

Simple flooding (SF) and basic definitions. We assume that the network contains only one information collection point called *base station*. To maintain the routing tree, the base station periodically floods the network with routing messages called *beacons*. *Epoch* is the period between floods. The epoch duration is selected long enough for two consecutive floods not to collide but short enough for the routing topology information to remain fresh. The base station initiates the flooding by broadcasting a *beacon* message. When a node receives a beacon for the first time in the epoch, it selects the sender to be its parent and rebroadcasts the beacon. The node ignores beacons for the rest of the epoch.

The following two algorithms preserve the mechanism of message forwarding used in simple flooding but alter the way each node selects its parent. Each node keeps the information about past beacon receipts and uses *simple moving average with a threshold* to improve parent selection. That is, each node averages the data from a fixed number (W) of past epochs. These epochs form a history *window*. The node selects the parent whose average value is the closest to the optimum. To make parent selection more stable across epochs, the node changes its parent only if the difference in the average values between the old parent and the new selection exceeds a certain *threshold* (T).

Average message count (AM). Unlike simple flooding, in AM each node records the identities of the neighbors from whom it receives beacons. Since every neighbor is supposed to send exactly one beacon per epoch, the node can calculate the message transmission rate for the W past epochs. The node selects its parent to be the neighbor with the highest transmission rate. The node changes the parent if there is another node whose transmission rate exceeds its parent's rate by more than T.

Average hop count (AH). In this algorithm a beacon carries the number of hops it travels. Each node that retransmits the beacon increments this number. Besides storing the identities of the senders, each receiving node also records the hop count for each beacon. On the basis of the stored information the node calculates the average hop count for W epochs for each individual neighbor and for the whole neighborhood. The node selects its parent to be the node whose individual hop count average is the closest to the neighborhood average. The node can change the parent only if the parent's average differs from the neighborhood average by more than T.

3 Experimental Platform, Setup and Metrics

Platform. As the architectural platform for our experiments we selected the MICA release of Berkeley's prototype networked sensors (*motes*) running TinyOS v. 0.6. Each such device [5] is equipped with a 4 MHz 8 Bit CPU, 128 KB instruction memory and 4KB RAM. For communication, the mote uses a 916MHz low-power single channel radio capable of delivering 10 KB/sec of raw bandwidth. The radio's range is between 10 and 25 feet. Each mote has an expansion slot that can accommodate a variety of sensors. TinyOS [4] is a lightweight event-based operating system that implements the networking stack, communication with the sensors and provides programming environment for this platform. TinyOS uses carrier sense multiple access medium access control. The nodes backs off for random amount of time in case a collision is detected. TOSSIM is a simulator that comes as a part of TinyOS release. It runs the code of TinyOS together with the application. TOSSIM simulates the motes' hardware as well as message transmission medium.

Simulation Setup. We simulated the setup of the original experiment [2]. In the experiment 156 motes were arranged on a plane in a 12 by 13 mote grid with a 2 feet grid spacing.

To simulate unreliable message transmission we modified TOSSIM to discard messages with probability depending on the distance between communi-

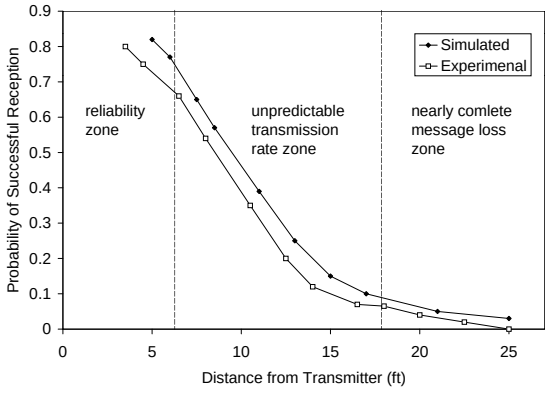

Figure 1: Probability of message reception over distance, experimental data is for "low" transmission power setting

cating nodes (see Figure 1). We selected the message loss probability according to the findings reported in the original study and elsewhere [1]. We discuss the fidelity of our simulation together with the rest of simulation results in the next section.

Metrics. We evaluate the effectiveness of a topology construction algorithm in two aspects: the topological quality of constructed trees and the algorithm's ability to adapt to changes in the environment.

To compare the quality of constructed trees we select the metrics that address the weaknesses of the simple flooding algorithm. Ganesan et al list the following quality problems in the simple flooding routing construction. *Long links* – links that are significantly longer than average (greater than 9 feet). Such links tend to be less reliable than shorter links. This problem is exacerbated by the way long links are used in routing. Long links tend to lead to areas not yet covered by the flood. In such areas, a lot of neighbors select the node connected by the long link as their parent. Thus, a long link is used more often than others. In addition long links have a greater chance of being asymmetric (i.e. the message loss rate of the link in one direction is significantly greater than that in the other). Notice, that asymmetric links selected by simple flooding tend to have lesser reliability in the direction of base station which leads to poor data transmission in that direction. *Backward links* – links leading towards the base station rather than away from it. Backward links unnecessarily increase the length of the path to the base station. *Stragglers* – nodes that do not select a parent. Hence, we select the number of long links, backward links and stragglers to be the quality measure of a routing tree.

The two adaptive properties of an algorithm is its stability and agility [7]. If an algorithm is *agile*, it quickly responds to changes in the environment. If an algorithm is *stable* it resists noise in individual observations. We selected the variation of hop count as a benchmark to compare the adaptive properties of the algorithms. To measure the agility we recorded the number of epochs it takes for the algorithm to arrive within the standard deviation of the average value of the hop counts for the nodes in the experiment. To measure stability we recorded the variations in the hop counts across epochs.

4 Analysis of Simulation Results

We started our experiments by verifying how close the results produced by simple flooding in the simulator match those of physical experiments reported earlier. We then varied the threshold size and the history window size for AH and AM to observe how this influences the adaptive properties of the algorithms. We selected the optimum values for these parameters and used them to compare the topological quality of the routing trees constructed by AH and AM as well as SF.

Fidelity Verification. The experiments by Ganesan et al [2] show that the message propagation pattern for Berkeley prototype sensors can be quite intricate. Choi et al [1] also studied the loss rates and message propagation for this platform. Their opinion is that the message propagation can be divided into

three general zones: the inner zone – reliable transmission; the middle zone – unpredictable loss rate, in this region the reception reliability of a node varies greatly and cannot be reliably predicted; the outer zone – almost complete message loss. We selected the reliability values for our modification to TOSSIM to follow the probabilities reported by Ganesan et al and such that the reception area is divided in three regions according to Choi et al (see Figure 1).

To verify that the selected simulation platform faithfully represents the physical experiments we reproduced the experiments of Ganesan et al for the simple flooding algorithm and compared the experimental and simulated data. We used the number of long links, back links and stragglers as a basis for comparison (see Figure 10). The numbers for experimental and simulated results are relatively close.

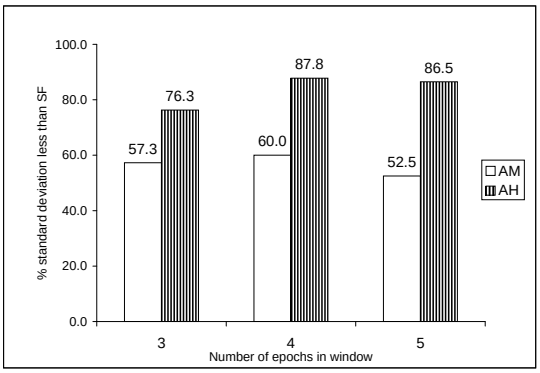

Figure 2: Standard deviation of average hop count over window size, threshold size is 0.4

Tuning parameters for adaptive properties. To measure the stability of the algorithms we calculated the standard deviation of average hop count for each node of the network. For that we ran SF and AM 10 times and AH – 5 times (AH takes somewhat longer to run). Each run of SF contained 1 epoch and each run of AM and AH – 5 epochs. For each node we computed the hop count at the end of each run and averaged it across the runs. We then computed the standard deviation for each node for AM, AH and SF. We computed the percentile of the nodes whose standard deviation for AM (AH) is lower than that for SF. The larger the percentile the more stable the algorithm is. To establish optimum stability parameters we repeated the experiments for various threshold and window sizes. The results of the experiments are shown in Figures 2 and 3.

In Figures 4 and 5 we plot standard deviation over distance from the base station. AH tends to exhibit less variations in standard deviation across all distances while AM tends to be more stable closer to the base station and less so on the fringes of the network.

Our experiments indicate that AH significantly outperforms AM. We believe that the stability of AH is fundamentally better than AM due to greater amount of information conveyed by beacons with hop counts. This information allows individual nodes to

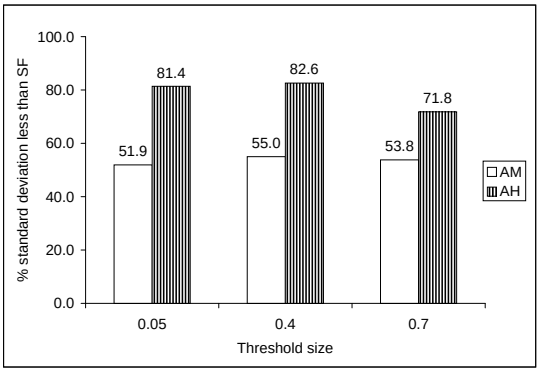

Figure 3: Standard deviation of average hop count over threshold size, window size is 4 epochs

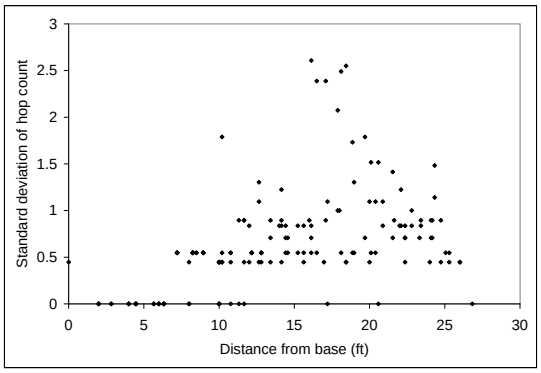

Figure 4: AM, Standard deviation of average hop count over distance

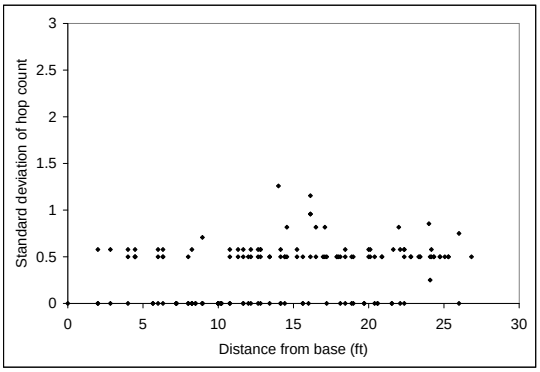

Figure 5: AH, Standard deviation of average hop count over distance

make more precise decisions in their parent selection.

To measure agility we selected 30 nodes on the periphery of the network at approximately the same distance from the base station (between 20 and 22 feet). We calculated the average hop count and standard deviation for this set of nodes over a large number of epochs. Then we observed how the average hop count for the set of nodes approached the computed value with each subsequent epoch. The results of the experiment for AH and AM are shown in Figures 6 and 7 respectively. In both cases the algorithms start within the boundaries of the standard deviation of the average value. This indicates good agility properties.

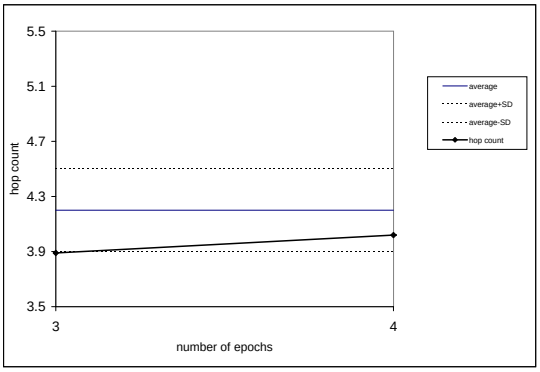

Figure 6: AH, average hop count of the periphery nodes over number of epochs

Topological quality comparison. For quality comparison we selected the history window size to be 4 and the threshold size to be 0.4. To illustrate the qualitative improvement of history based algorithms over SF we show an example routing topology constructed by SF and AM in Figures 8 and 9 respectively.

The average number of long links, back links and stragglers for each algorithm is shown in Figure 10. The results indicate that AH and AM show significant decrease in the number of backward links with respect to SF, no improvement in the number of long

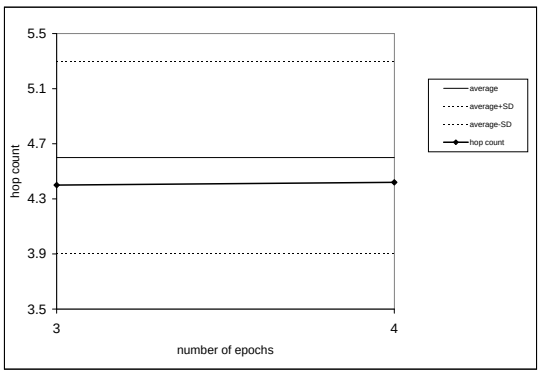

Figure 7: AM, average hop count of the periphery nodes over number of epochs

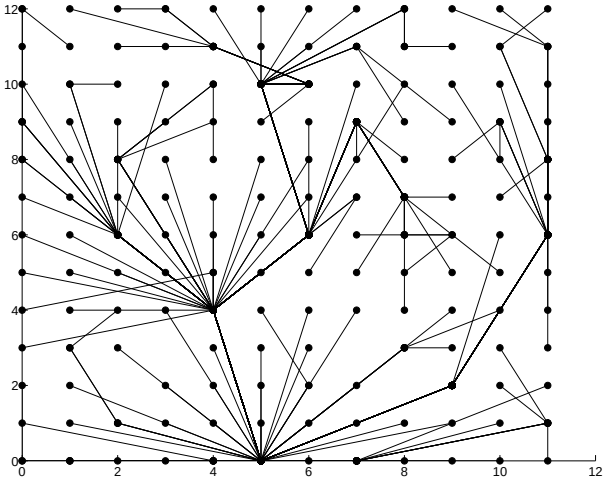

Figure 8: Example topology for simulated simple flooding (SF) parent selection

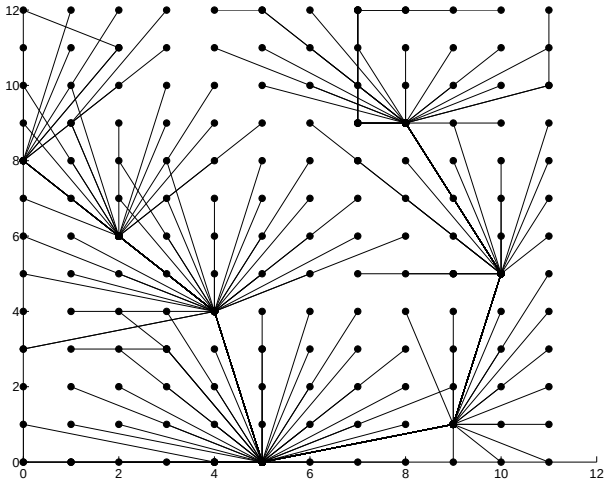

Figure 9: Example topology for average message count (AM) parent selection

links and little improvement in the number of stragglers.

Summary. We observed significant gains in stability in the average hop count algorithm (AH) and improvement in the number of backward links in the average message count algorithm (AM). Overall, the history based algorithms that we studied achieve stability of the topology without sacrificing quality or agility.

5 Conclusion

The algorithms presented in this paper can be extended in various ways.

Extensions to other metrics. Other quality metrics can be incorporated in our algorithms. Notice how AH uses the beacon to carry cumulative information about extended area of the network. This technique can be used, for example, to optimize power consumption in routing. If each sender appends its remaining power level to the beacon, the receiver can use this information for parent selection.

A problem of large clusters [2] can be addressed

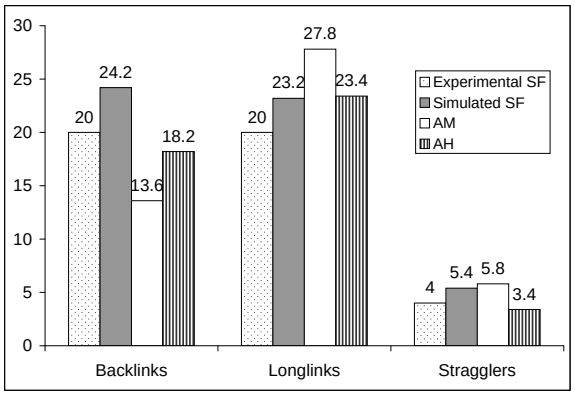

Figure 10: Comparison of topological quality parameters

similarly. Due to unpredictability of flooding, a large number of neighbors may select the same node to be their parent. This node will have to forward its children's messages toward the base station. This will potentially create a bottleneck and drain the node of its power.

Nodes connected by long links frequently become the heads of large clusters. This happens because long links tend to lead to areas not yet covered by the flood. Since long links are less reliable, the problem of large clusters is particularly acute. To contain the problem each node (a) needs to learn the number of its children, (b) propagate this information so that it is used in parent selection. We can consider a number of techniques for a node to learn the identities of its children. Once the node knows its cluster size, the node can include this information in the beacon. The receiver node can use the data about its neighbors' cluster sizes to help in parent selection.

Other extensions. Our algorithms used simple average, but more sophisticated techniques [7] such as window-based exponential moving average can be incorporated. Moreover, multiple metrics such as power and average path length can be used for parent selection simultaneously.

In this paper we assumed that there is a single base station in the network. However, our algorithms can be easily extended to multiple base stations. In this case the neighborhood information collected by each node can be reused.

Notice that our algorithms do not depend on the relative timing of the epochs and thus could be used in on-demand routing algorithms such as AODV, DSR or other algorithms that use flooding.

Further research objectives. We would like to study the fault tolerance properties of history-based flooding algorithms. In particular we would like to investigate how historical information can be leveraged to allow the algorithm to quickly recover both from single node failures and arbitrary state corruptions that model massive faults [9].

Another attractive research objective is to minimize storage requirements at the node. Currently the node has to collect data about its every neighbor. In dense networks this can become rather expensive. We would like to devise an extension to our algorithms where the amount of information about each neighbor is stored according to the possibility of this neighbor becoming the parent. Hence, the neighbors that do not have good chances of being used do not waste the node's memory.

6 Acknowledgments

We would like to thank Anish Arora of Ohio State University for his ideas, comments and helpful discussions.

References

[1] Y. Choi, M.G. Gouda, M.C. Kim, and A. Arora. The mote connectivity protocol. Technical Report TR03-08, University of Texas at Austin, 2003.

[2] D. Ganesan, B. Krishnamachari, A. Woo, D. Culler, D. Estrin, and S. Wicker. Complex behavior at scale: An experimental study of low-

power wireless sensor networks. Technical Report CSD-TR 02-0013, UCLA, 2002.

[3] Z. Haas, J.Y. Halpern, and L. Li. Gossip-based ad hoc routing. In *Proceedings of 21st Conference on Computer Communications (InfoCom)*, New York, June 2002. IEEE.

[4] J. Hill, R. Szewczyk, A. Woo, D. Culler, S. Hollar, and K. Pister. System architecture directions for networked sensors. *ACM SIGPLAN Notices*, 35(11):93–104, November 2000.

[5] J.L. Hill and D.E. Culler. Mica: A wireless platform for deeply embedded networks. *IEEE Micro*, 22(6):12–24, November/December 2002.

[6] D.B. Johnson and D.A. Maltz. Dynamic source routing in ad hoc wireless networks. In T. Imielinski and H. Korth, editors, *Mobile Computing*, volume 353. Kluwer Academic Publishers, 1996.

[7] M. Kim and B. Noble. Mobile network estimation. In *Proceedings of the Seventh Annual International Conference on Mobile Computing and Networking (MobiCom-01)*, pages 298–309, New York, July 16–21 2001. ACM Press.

[8] Y.B. Ko and N.H. Vaidya. Location-aided routing (LAR) in mobile ad hoc networks. In *Proceedings of the 4th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom-98)*, pages 66–75, New York, October 25–30 1998. ACM Press.

[9] M. Nesterenko and A. Arora. Self-stabilizing routing in wireless embedded systems. In *Proceedings of SRDS Workshop on Reliability in Embedded Systems*, pages 16–22, New Orleans, Louisiana, October 2001.

[10] S.Y. Ni, Y.C. Tseng, Y.S. Chen, and J.P. Shen. The broadcast storm problem in a mobile ad hoc network. In *Proceedings of the Fifth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom-99)*, pages 151–162, N.Y., August 15–20 1999. ACM Press.

[11] C.E. Perkins and E.M. Royer. Ad-hoc on-demand distance vector routing. In *Poceedings of the Second Annual IEEE Workshop on Mobile Computing Systems and Applications*, pages 90–100, February 1999.