Real-World Experiences with an Interactive Ad Hoc Sensor Network

Mark D. Yarvis, W. Steven Conner, Lakshman Krishnamurthy, Jasmeet Chhabra, and Brent Elliott
Intel Labs
{mark.d.yarvis, w.steven.conner, lakshman.krishnamurthy}@intel.com
{jasmeet.chhabra, brent.j.elliott}@intel.com

Alan Mainwaring
Intel Research
amm@intel-research.net

Abstract

While it is often suggested that moderate-scale ad hoc sensor networks are a promising approach to solving real-world problems, most evaluations of sensor network protocols have focused on simulation, rather than real-world, experiments. In addition, most experimental results have been obtained in limited scale. This paper describes a practical application of moderate-scale ad hoc sensor networks. We explore several techniques for reducing packet loss, including quality-based routing and passive acknowledgment, and present an empirical evaluation of the effect of these techniques on packet loss and data freshness.

1. Introduction

Sensor networks provide a promising mechanism for mining information from the physical world. Steady improvements in IC and radio technologies continue to shrink sensor devices, making large-scale sensor networks feasible. A great deal of research attention has been placed on the development of protocols for organizing ad hoc sensor networks, efficiently obtaining relevant information from a network, and ensuring network longevity. Researchers have tended to evaluate these protocols through simulation and small-scale testing. The feasibility of ad hoc sensor networking in moderate scale and under real-world conditions remains an important issue that is largely unexplored.

One application of sensor networks is to enable audience feedback of a lecture or presentation. Each audience member is given a voting device (Figure 1)—a small box with buttons for input and LEDs for feedback[1], powered by a version of the Berkeley mote [5]. The network allows users to respond to questions by pressing one of the buttons and determines the number of users that pressed each button. Users may change their response over time, requiring a continuous tally.

Figure 1: External and internal views of a mote-based voting device.

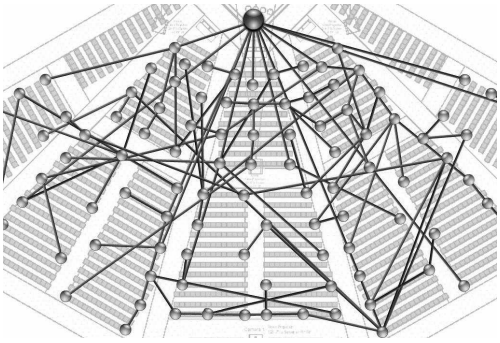

Figure 2: A 100 node ad hoc topology of handheld voting devices deployed in the San Francisco Moscone Convention Center.

While this application could be enabled by a fixed infrastructure built into a particular auditorium, a wireless network could enable audience participation in any venue. Since engineering a wireless network that completely covers a particular venue can be time consuming and impractical, a multihop ad hoc network is ideal. To this end, we constructed a wireless sensor network of over 100 nodes (Figure 2). Nodes in the network delivered results via a designated gateway node over an ad hoc routing topology which was generated using a single-destination implementation of the DSDV protocol [12].

¤ The authors wish to thank Steven Fordyce for his help constructing and maintaining the voting devices.
† Other names and brands may be claimed as the property of others.
[1] In the future, a more sophisticated device could be built using gravity switches or other sensors, allowing the speaker to obtain a "sense of the audience."

Even at this moderate scale, we found that obtaining reasonable performance in this network was a challenge. The main impediments were congestion and packet loss. Our application requires information from all nodes to be simultaneously collected at a single location, resulting in data implosion. We leveraged several techniques, including packet aggregation, to reduce the amount of data converging on the gateway. In addition, we found that packet loss, both during transmission and within intermediate nodes, was difficult to control given the simplistic radios and limited resources of the nodes. We explored techniques at both the link layer and the network layer to reduce packet loss. This paper describes our real-world experiences with moderate sized ad hoc sensor networks, and techniques that can be employed to reduce congestion and packet loss in these networks, including our experimental results.

2. The application architecture

The voting application consists of four parts: a sensor network of voting nodes, a gateway node with an interface to a traditional computing platform, a sensor gateway application, and one or more client applications. This architecture is shown in Figure 3.

The voting nodes are built upon a variant of the rene mote [5], originally developed at the University of California at Berkeley, which serves as the base communication and computation platform. The core components of the mote are an Atmel® ATmega163L AVR® microcontroller [1] with 16 KB of flash, 1 KB of RAM, and an RFM TR1000 916 MHz radio transceiver [14]. Protocols and applications are implemented using TinyOS [5], an event-driven operating system designed to fit within the minimal resources of mote hardware.

The radio provides on-off keying modulation and delivers a raw bit rate of 10 kbps. SEC/DED (single error correcting, double error detecting) baseband encoding is implemented in software, resulting in a maximum theoretical channel capacity of approximately 568 bytes per second. Because each packet is 37 bytes long, the channel capacity has a theoretical limit of 15 packets per second. Channel access is controlled by carrier sense multiple access with collision avoidance (CSMA/CA).

One node in the network acts as a gateway, allowing results to be obtained from the sensor network. Each node is capable of periodically sending the user's vote to the gateway, where it can be delivered, across a serial link, to a "sensor gateway" process on a PC. Since the user can change his vote during the voting process, and since packets can be lost in transport, each voting node sends the user's current vote once per time interval. As voting packets are passed hop-by-hop across the sensor network, each node attaches its ID to the packet, forming a "traceroute."

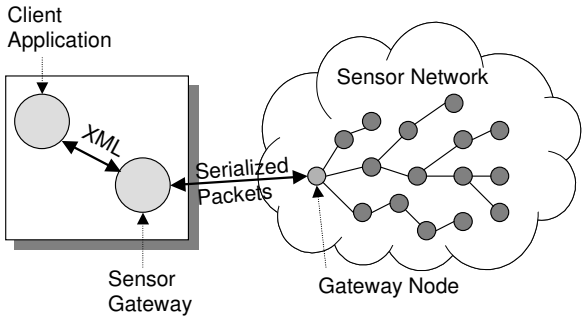

Figure 3: Software architecture of the voting application.

The sensor network forms an ad hoc topology using a single-destination form of the DSDV protocol [12]. In this case, the gateway initiates DSDV route requests, which are flooded into the network. Each route request contains a monotonically increasing sequence number, a metric, and the identity of the previous hop. The metric is a monotonically increasing measure of the cost of transmission to the destination. A typical metric is hop count, allowing the shortest path to be identified. We will explore disadvantages to this metric as well as a link-quality-based alternative in Section 4. By optimizing the metric and discarding any packets with outdated sequence numbers, a node can identify the best next hop along a path to the gateway node. Together, these paths form a routing tree rooted at the gateway.

The gateway node delivers the packets it receives to the "sensor gateway" process running on a PC. This process keeps track of the current total vote. Using the traceroute in the vote packets, the gateway process also maintains a representation of the topology of the sensor network. The gateway process serves the current vote and network topology in an XML format over a TCP socket to client applications, such as the display application shown in Figure 2.

3. Traffic reduction

The voting application illustrates a fundamental challenge for sensor networks: the volume of sensor data can readily over-commit the available system resources. In particular, network bandwidth is an extremely limited resource in such networks.

If each node in a 100 node network were to send its vote to the gateway once every 20 seconds, the gateway would receive 5 vote packets each second. While the channel supports a theoretical maximum of 15 packets per second, contention reduces the achievable channel capacity in an ad hoc network to a fraction of the actual channel capacity [9]. In addition, DSDV route update packets are also being transmitted once per node every ten seconds. As a result, this application uses a signifi-

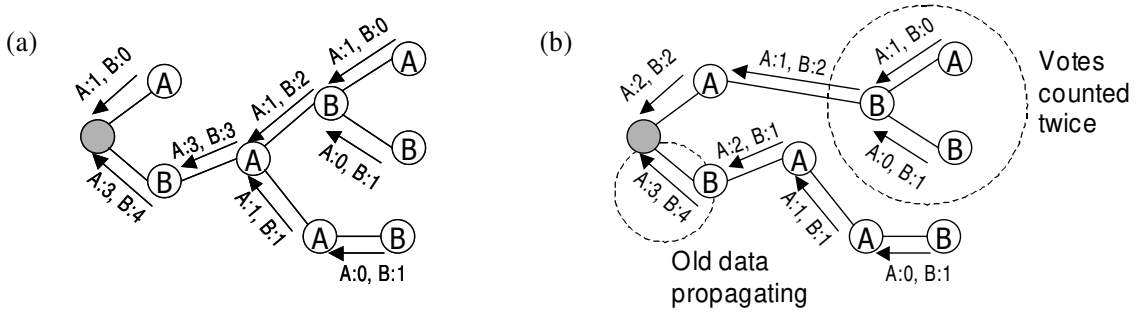

Figure 4: Node motion can result in unbounded aggregation errors.

cant amount of the available bandwidth. Reducing the number of packets delivered to the gateway would likely be beneficial.

While aggregation is a natural solution to this problem, vote aggregation is challenging because topology changes can result in unbounded errors. Consider the network in Figure 4a. During each local time interval, a node waits for its childrens' votes. At the end of an interval, the node forwards aggregated vote totals (including its own vote) to its parent. Thus, it takes *n* time intervals for a vote from a node at tree-depth *n* to be received by the gateway. If a subtree moves (due to a topology change) to a higher position in the tree (Figure 4b), its votes would be double counted for one or more time intervals (depending on the distance moved). Since a subtree can consist of any number of nodes, the resulting error is unbounded. Similarly, if a subtree moves deeper in the tree, its votes will not be counted for one or more intervals. In addition, if packet retransmission is used to reduce packet loss, any topology changes can result in aggregation errors, even if a node's depth in the tree is unchanged. Note that while this problem is not present in min/max aggregation functions, these functions are not applicable to vote aggregation.

An alternative approach is to aggregate packets, rather than aggregating data. A vote from a given node requires a single byte to identify the voting node and two bits to specify the vote (A, B, or no vote). Each leaf node on the tree can generate a vote packet containing its own identity and its vote, and forward it toward the root of the routing tree. When forwarding a vote packet, each node can also append its own identity and vote. Because the gateway can distinguish the source of each vote, it can maintain the last known vote from each node and provide the current vote count to client applications.

Since the above aggregation scheme only requires leaf nodes to initiate packets, many fewer packets need be delivered to the gateway. In our experiments, an average of 55% (σ = 5%) of nodes were leaf nodes. Leaf node detection can be implemented using a timeout; each node transmits a vote packet if it has not forwarded a vote packet (appending its own vote) over some time period. Note that despite reducing the network traffic, this aggregation scheme increases redundancy; votes from non-leaf nodes can have multiple opportunities to be received by the gateway. In addition, since a node need not wait for its childrens' votes before forwarding data to its parents, the latency involved in reporting votes is reduced.

The amount of aggregation is limited by packet size; in our system 13 votes. When a packet becomes full, it can be forwarded to the gateway. A node that detects a full packet behaves as a leaf node.

4. Quality-based routing

Hop count is a typical metric used to identify an optimal path to the destination. However, in a wireless network hop count is potentially a bad choice. Link quality between pairs of nodes may vary during the lifetime of a network based on distance, transmit power, antenna shape and orientation, radio interference, and environmental factors (such as people in the sensor network field attenuating radio signals). Moreover, such variation may lead to asymmetric links between nodes in the network. Even if the locations of nodes in the network are fixed and each node is configured with an identical transmit power, node interconnectivity will change during the course of an experiment.

Given these operating conditions, blindly selecting DSDV parents based on minimum number of hops to the destination may result in poor route choices. For example, a node that is typically 7 hops from the destination may sporadically receive a route update (*RUPDATE*) message from a node that is 4 hops from the destination. However, sending data packets along this route may result in greater packet loss, since packet losses are higher over extreme physical distances.

In an attempt to alleviate these drawbacks, our DSDV implementation uses a link cost metric that is based on link quality statistics.

4.1. Measuring link quality

To facilitate link quality tracking, nodes tag each outgoing packet with a one-hop sequence number, independent of packet type. Nodes track these one-hop sequence numbers in packets received from each of their neighbors. Any sequence number gaps identified at a receiver indicate packets that were sent but not received. Each node learns the downstream reliability from each of its neighbors by storing the number of packets successfully received out of the last 32 packets sent.

To identify bi-directional quality, nodes share their local quality statistics with each of their neighbors. This data could be transmitted in a new periodic message. However, to avoid consuming additional bandwidth, we chose to piggyback these statistics onto the *RUPDATE* messages that are already periodically broadcast from each node for DSDV route establishment. Due to the limited space in *RUPDATE* messages, neighbor lists are divided into four categories based on quality thresholds: Q_3 is 0%-10% loss, Q_2 is 10%-21% loss, Q_1 is 21%-53% loss, and Q_0 is 53%-100% loss. When a node receives a quality list from a neighbor, it supplements its local quality statistics for that neighbor with the neighbor's perception of the quality of the link from itself. Bi-directional link quality is considered to be the minimum quality value in each direction between a pair of nodes. If a node is not listed in its neighbor's quality list, the link is assumed to be asymmetric and receives the minimum quality rating of Q_0.

There are several notable limitations to the link quality implementation, primarily relating to neighbor storage (1 KB of total data space in our platform) and communication constraints (31 bytes of payload space). In a dense network, a node may have more neighbors than it can track, and poor parent selection may result. When the neighbor list is full and a new neighbor is detected, the current neighbor list is searched, and the neighbor with the lowest link quality is replaced. If a node is not listed on its neighbor's neighbor list, the node must assume that an asymmetric link exists. This is normally the intended behavior, but if it is missing from the list due to neighbor overflow then a high-quality neighbor may be overlooked.

The initial quality value for a new neighbor must be chosen carefully. If the initial value is set too low, identification of a high quality neighbor will require a long delay. On the other hand, an initial value that is too high may result in replacing a higher quality neighbor in the neighbor table if the table is full due to limited storage space, potentially resulting in table thrash. In the current network implementation, the initial quality value is chosen to be 45% loss, near the midpoint of Q_1.

4.2. Using link quality as a routing metric

We have integrated link quality metrics with DSDV route selection. In particular, we wanted to allow DSDV to select routes with the highest end-to-end packet delivery rate.

The end-to-end packet delivery rate of a given route can be computed by multiplying the delivery rate of each link (measured as described in Section 4.1) along that route. This approach is impractical because floating-point multiplication is expensive in terms both of code size and storage complexity. In addition, our link quality implementation can only provide an estimate of link quality by categorizing each link into one of four quality categories. As a result, we have chosen to convert the quality measure of each link into a link cost, where higher link costs are assigned to links that have a higher estimated loss rate. Link costs along a path can be summed, producing a monotonically increasing metric for the DSDV algorithm that identifies paths with the highest packet delivery rate.

Appropriate link costs can be chosen by converting packet delivery rates to the log scale and then normalizing to the integer domain. Because the link metric is in the log domain, adding link metrics is equivalent to multiplying packet delivery rates, allowing paths of different lengths to be correctly compared, without requiring expensive multiplication.

As described in Section 4.1, each node maintains an estimate of the quality of the link to each neighbor. The estimate corresponds to a range of packet delivery rates. The metric for each link is determined by normalizing the log of the estimate value for each category, as shown in Table 1. Because there are a small number of categories, the associated link costs can be precomputed.

Upon receipt of an *RUPDATE* message with a new sequence number, each node determines the quality of the link to the sender, selects the metric associated with that link quality, and adds the link metric to the route cost in the *RUPDATE* message. This metric is then used to identify the lowest cost, and thus highest delivery rate, path to the sink node. The node can then advertise its own route cost, based on the lowest cost identified. Note that due to limited packet size, our implementation restricts the total

Table 1: Link metrics associated with each level of link quality.

Quality	Delivery Rate	Delivery Rate Estimate (R_e)	$\ln(R_e)$	Link Metric
Q_3	90-100%	0.95	-0.05	1
Q_2	79-90%	0.85	-0.16	3
Q_1	47-79%	0.65	-0.43	8
Q_0	0-47%	0.25	-1.39	28

route cost to be less than 255. To allow for networks of reasonable size, link metrics of 6 and 15 were used for Q_1 and Q_0 instead of the theoretically correct values.

Ideally, each node should wait for a minimum "settling time," the expected delay from the time a node receives its first *RUPDATE* message with a new sequence number and the time it receives the *RUPDATE* message for its best parent, before selecting its parent [12]. Due to code space limitations, settling time is not currently implemented in the network. Instead each node waits a random interval after receiving an *RUPDATE* message with a new sequence number before advertising its own lowest delivery cost. While not perfect, this scheme does reduce the overhead associated with *RUPDATE* messages. In addition, randomization has been shown to reduce the inherently bursty nature of traffic in sensor networks [15].

5. Empirical results

To evaluate the performance of ad hoc networking protocols in our voting application, we deployed the voting nodes in regular topologies of 24 (4x6), 48 (6x8), and 91 (7x13) nodes, with a gateway at one end (in the middle of one of the short edges). In each case, nodes were deployed in an office environment over a rectangular grid with nodes 4 feet apart in one direction and 6 feet apart in the other. The radio range of the nodes was adjusted such that, in a noiseless environment, nearly lossless communication was possible at a distance of up to 10 feet. Each experiment was run for approximately one hour, producing 1000 to 3000 packets at the gateway. In all experiments, vote aggregation (as described in Section 3) was used to reduce data implosion at the gateway node, with leaf nodes generating vote packets every 20 seconds.

In all experiments, DSDV was used with a route update interval of 10 seconds. Three flavors of DSDV were compared: straight DSDV, DSDV with asymmetric link detection, and DSDV with link quality monitoring. In the base case, DSDV uses a hop count as the sole metric. In the second case, nodes exchange neighbor lists (described in Section 4.1) to identify and avoid asymmetric links when selecting a parent node. In the third case, nodes maintain a recent history of the link quality to their neighbors and use bi-directional link quality to select the parent with the lowest cost route to the gateway (described in Section 4).

Asymmetric link detection was implemented using link quality monitoring by setting the quality thresholds so that each link was forced into one of two categories, depending on whether any packets had been received from that neighbor. To provide straight DSDV, link quality monitoring was disabled by setting the thresholds so that all nodes were forced into a single category.

All results are shown with 90% confidence intervals.

5.1. Loss rates

Figures 5 through 9 show the end-to-end packet loss measured for the three network sizes. Loss rates are computed using an end-to-end sequence number maintained by each leaf node. Packet losses are identified by gaps in the sequence numbers of packets generated by a given leaf node. Loss rates are broken down by node depth. Packets that are successfully received are attributed to the number of hops recorded in the packet. Since the network is dynamically changing, a given node can change its location and depth in the DSDV tree over time. Since it is difficult to accurately attribute a node depth to a dropped packet, we have chosen to attribute each lost packet to a node's previously known tree depth.[2]

In the 24-node experiment (Figure 5), use of the straight DSDV algorithm resulted in moderate packet loss. As expected, packet loss was greater for nodes that were further (in terms of hops) from the gateway. Both asymmetric link detection and link quality monitoring reduced packet losses. In particular, link quality monitoring reduced packet losses by between 24 and 32%, except in the case of nodes 1 hop from the gateway as discussed in Section 5.2.

The 48-node experiment (Figure 7) resulted in similar, but less dramatic results. In this case, link quality monitoring reduced packet loss by between 6 and 20%. The 91-node experiment (Figure 9) produced an improvement of only 2 to 4%.

From these results, it appears that both quality-based routing and asymmetric link detection become less effective at reducing the packet loss rate as the size of the network increases. We have identified two likely causes. First, the communication channel may be reaching capacity, particularly near the gateway. This theory is supported by the relatively flat loss rate for nodes beyond 2 hops, suggesting that a majority of losses occur near the gateway. Second, the neighbor list may be overflowing, causing it to be ineffective (as described in Section 4.1).

Under the assumption that channel contention was an issue, we augmented our MAC layer to include a passive acknowledgment scheme [7] in which a node retransmits a packet if it does not overhear its parent forwarding that packet. Our implementation provides best-effort retransmission in that (due to platform limitations) only one packet can be buffered at a time. In addition, retransmissions never occur from or to the endpoints.

The results of 24- and 48-node experiments that included passive acknowledgments are shown in Figure 6 and Figure 8. Note that in the 24-node case, passive ac-

[2] Had we instead chosen to freeze the network topology by stopping the DSDV protocol, the amount of network traffic would have been altered and the topology would have been unable to adapt to changing link conditions.

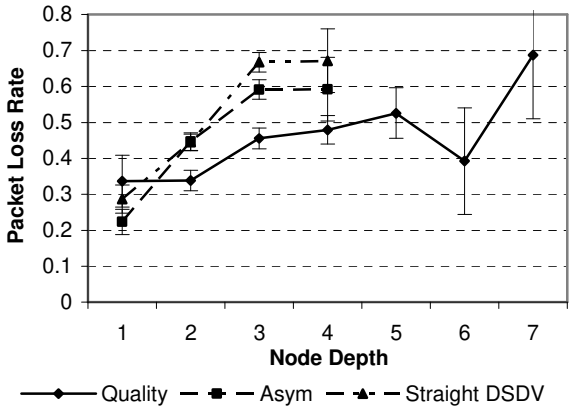

Figure 5: Packet loss rates in a 24-node network with straight DSDV, DSDV with link quality detection, and DSDV with asymmetric link detection.

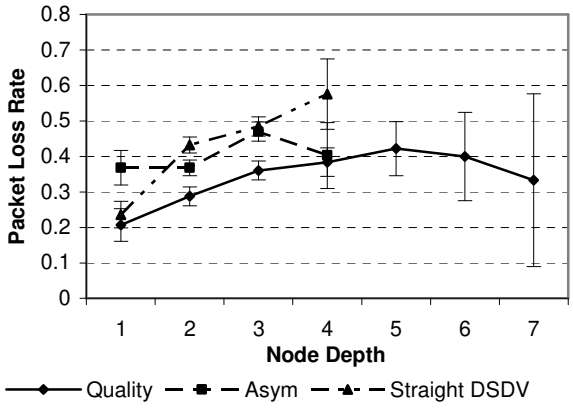

Figure 6: Packet loss rates in a 24-node network under three DSDV variants, with the addition of passive acknowledgments.

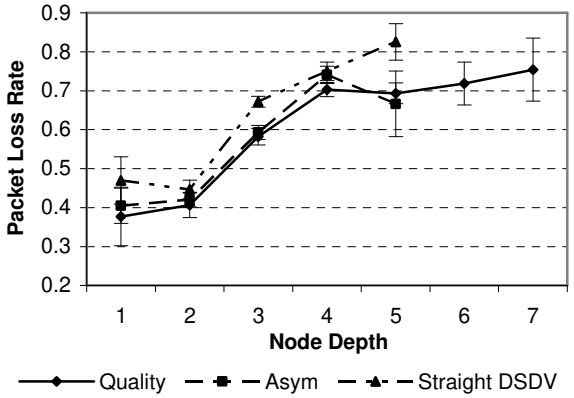

Figure 7: Packet loss rates in a 48-node network with straight DSDV, DSDV with link quality detection, and DSDV with asymmetric link detection.

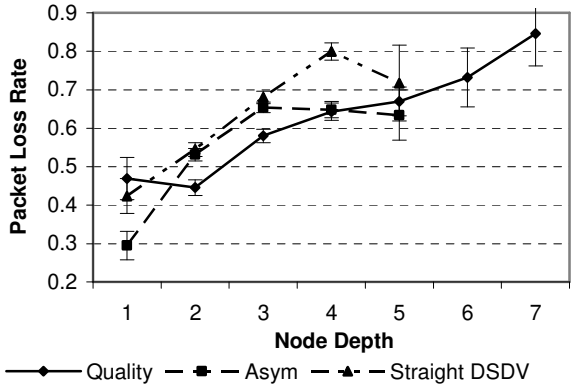

Figure 8: Packet loss rates in a 48-node network under three DSDV variants, with the addition of passive acknowledgments.

knowledgments reduced the overall loss rates by roughly 25%. However, loss rates in the 48-node case and in a 91-node experiment (not pictured) were actually increased by passive acknowledgments. We believe the negative effect of passive acknowledgment in the larger experiments resulted because our DSDV implementation does not quell duplicate packets produced by the MAC layer. As a result, any unnecessary retransmission (caused when a node fails to see its neighbor forward a packet) will produce a duplicate data packet that will be forwarded to the gateway. The increased traffic has a negative effect on larger networks which are already congested, offsetting possible benefit. We believe that if duplicate packets were discarded within the network, rather than being forwarded, passive acknowledgment could improve loss rates in larger networks. It is worth noting that, in the 48-node experiments (Figure 7 and Figure 8), while loss rates for straight DSDV rose, loss rates for quality-based routing did not. Quality-based routing may help to route around the additional congestion caused by the lack of in-network duplicate detection.

5.2. Problems with loss rate

As noted above, in several of our experiments quality-based routing appeared to increase packet loss rates for nodes within one or two hops of the gateway. This result is a side effect of our measurement methodology. As described in Section 3, packets originate only at leaf nodes. Since loss rates reported in this section are measured using an end-to-end sequence number (link-level sequence numbers are only available to nodes within the network), loss rates can only be measured for leaf nodes. However, with our quality-based routing algorithm, a node becomes a leaf node because it advertises (to its neighbors) a relatively poor route to the sink node. As a

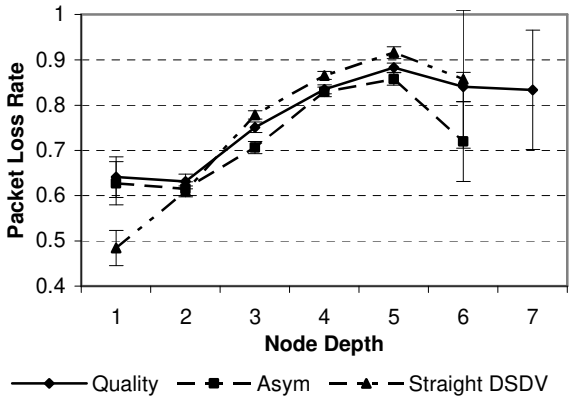

Figure 9: Packet loss rates in a 91-node network with straight DSDV, DSDV with link quality detection, and DSDV with asymmetric link detection.

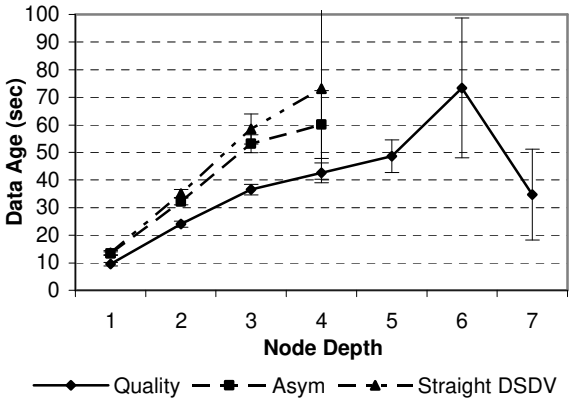

Figure 10: Data age in a 24-node network with straight DSDV, DSDV with link quality detection, and DSDV with asymmetric link detection.

result, packet loss rates will tend to be inflated in the case of quality-based routing, particularly for nodes near the gateway.

This problem could be eliminated if each node attached its own sequence number while forwarding a packet, allowing average loss rate to be based on both interior nodes and leaf nodes. This approach was not possible in our application, due to limited packet size. Instead, the following subsection describes an alternative metric, *data age*, which can be more easily measured across all network nodes.

5.3. Data age

Data age is the amount of time between subsequent votes received from a particular node. Data age is a better metric for this application because it can be measured for all nodes, not just leaf nodes. In addition, data age measures a quality of the network that is visible to the user: the latency between the time a vote is cast and the time it is reflected in the results. While leaf nodes generate a data packet every 20 seconds, data age for interior nodes can be significantly less, because these nodes add their data to all forwarded packets. On the other hand, data age can also be significantly increased by packet loss.

Average data age in a 24-node network is depicted in Figure 10. Note that quality-based routing reduced the data age by between 30 and 42%. While quality-based routing tends to increase the tree depth (because the shortest path is not necessarily the best path), data originating from deep in the quality-based routing tree did not tend to be significantly older than data originating deep in the straight DSDV routing tree.

Average data ages for 48- and 91-node networks are depicted in Figure 11 and Figure 12 respectively. In each

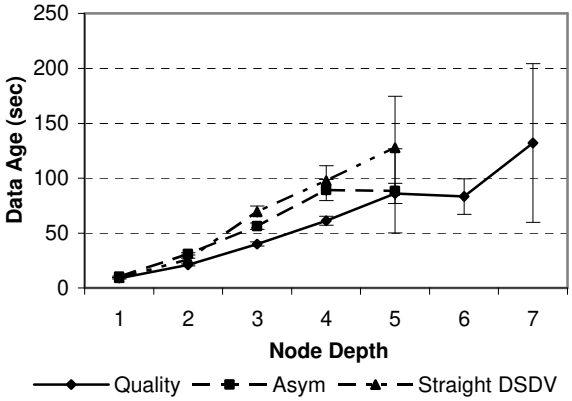

Figure 11: Data age in a 48-node network with straight DSDV, DSDV with link quality detection, and DSDV with asymmetric link detection.

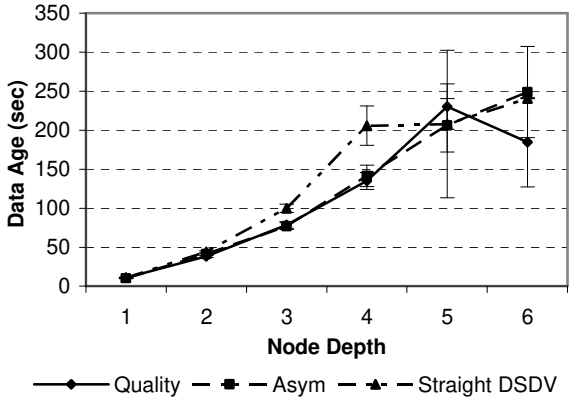

Figure 12: Data age in a 91-node network with straight DSDV, DSDV with link quality detection, and DSDV with asymmetric link detection.

case, the improvement in data age is less than in the 24-node case, but still significant. The diminishing benefit in data age is most likely linked to the increased packet loss in larger networks described in Section 5.1. Despite these packet losses, quality-based routing allowed the average age of data to be reduced by as much as 42% in the 48-node network and 34% in the 91-node network, significantly improving network responsiveness.

6. Related work

Published prior work evaluating ad hoc networking protocols has mainly focused on simulating networks of around 50 nodes. Given the lack of implementation results, we compare our empirical evaluation with simulation results. The effect of mobility and protocol routing overhead has been studied extensively with simulation [2,4]. The lessons learned are not directly applicable in our environment because of differences in both the kind of network that was simulated (802.11) and the typical workload (mobile nodes). It would be useful to understand the effect of the DSDV route update interval for a given mobility rate (or fixed), versus network size and transmission workload. A study of a multihop wireless ad hoc network testbed [11] reports the effect of mobility on the performance of TCP/IP. Our application domain is quite different from the mobile IP network studied in this paper. Moreover the network included only 5 mobile nodes and 2 fixed nodes.

Evaluation of a transmission control scheme for sensor networks using both simulation and Berkeley motes has been presented in [15]. The evaluation uses 11 nodes in a fixed network topology with a network depth of 5 hops. The paper defines the delivery bandwidth at the gateway and energy efficiency as metrics for evaluating sensor network protocols. This research showed that random transmission delays and random back-off in the MAC layer could remove periodic contention. Our implementation used the same MAC layer presented in this work and incorporated back-off techniques to improve network performance. Our evaluation expands on previous results by exploring the impact of scale as well as integrating other performance-enhancing techniques.

While many ad hoc routing protocols assume link symmetry, other researchers have identified that asymmetric links are a problem in wireless ad hoc networks. Asymmetry has been dealt with in a variety of ways, including the exchange of neighbor lists in periodic local messages to detect asymmetric links [13], similar to the approach used in our network. End-to-end techniques have also been employed to deal with asymmetric links, for example, by flooding route reply messages rather than simply reversing the route request path [8]. Likewise, techniques have been described to measure local link quality by detecting packet loss for the purpose of detecting and avoiding neighbors with high loss rates [3]. Our network extends this approach by piggybacking link quality information with exchanged neighbor list messages, allowing bi-directional link quality to be estimated (Section 4.1).

A taxonomy of aggregation schemes has been presented in [10] which classifies aggregates based on attributes important to sensor networks. Under this taxonomy, vote aggregation falls under the *distributive* aggregate classification, theoretically allowing partial state records to be no larger than the final aggregate size. However, frequent route changes in our ad hoc voting network provided many opportunities for vote duplication (Section 3), to which distributive aggregation techniques are very sensitive. This resulted in the implementation of a *holistic* aggregate approach, also described in the taxonomy, where individual votes were piggybacked on traceroute packets to be counted at the gateway, precluding duplicate counting. Other researchers have also noted that such *packing aggregation* saves energy and bandwidth by reducing the total per-transmission overhead in the network [6].

7. Next steps

The results described in Section 5 represent an initial rather than comprehensive exploration of the performance of an ad hoc sensor network. Many issues remain outstanding that we plan to explore in the future.

In this paper, we characterized the network in terms of packet loss. Other metrics are also possible. In particular, the shape of the routing topology would be of interest. We have noted that augmenting DSDV with asymmetric link detection or link quality monitoring tends to produce trees that have more balanced fan-in but also tend to be deeper. We would like to understand the effect of different metrics on the topology of a DSDV-routed network.

The results presented in this paper reflect only a few changes to the algorithmic parameters. It would be valuable to understand the best route update interval for this application, balancing bandwidth against resilience to change. In addition, we would like to understand how various quality thresholds affect packet loss and whether absolute thresholds could be replaced by a quartile ranking.

Finally, while we have evaluated several techniques for reducing packet loss, we would also like to identify the major sources of packet loss. Through further instrumentation of our code, we hope to identify the degree to which single-packet buffering, limited packet processing time, packet coding, and channel contention contribute to packet loss. In addition, we would like to determine where in the network packet loss is most likely to occur.

8. Conclusions

Empirical measurements provide valuable insights into the performance of ad hoc sensor network communication protocols in the context of real application environments. While perhaps only a first step, our results provide insight into the performance of the DSDV protocol in actual networks of moderate scale. In particular, we have shown that, while packet losses can be quite high, known techniques such as link quality monitoring and passive acknowledgment can produce measurable improvements in real networks. These results have allowed us to demonstrate the value of moderate-scale ad hoc networking applications. While there is more work to be done in this space, these initial results provide a promising start.

References

[1] Atmel Corp., "AVR Microcontroller ATmega163L Reference Manual," http://www.atmel.com/.

[2] J. Broch, D. A. Maltz, D. B. Johnson, Y. C. Hu, and J. Jetcheva, "A Performance Comparison of Multi Hop Wireless Ad-Hoc Network Routing Protocols," *Proceedings of the Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM '98)*, Dallas, TX, October 1998.

[3] Alberto Cerpa and Deborah Estrin, "ASCENT: Adaptive Self-Configuring Sensor Networks Topologies," *Proceedings of the Twenty First International Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2002)*, New York, NY, June 2002.

[4] Samir R. Das, Charles E. Perkins, Elizabeth M. Royer and Mahesh K. Marina, "Performance Comparison of Two On-demand Routing Protocols for Ad hoc Networks," *IEEE Personal Communications Magazine*, February 2001, p. 16-28.

[5] Jason Hill, Robert Szewczyk, Alec Woo, Seth Hollar, David Culler, and Kristofer Pister, "System Architecture Directions for Networked Sensors," *Proceedings of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems*, Cambridge, MA, November 2000.

[6] Chalermek Intanagonwiwat, Deborah Estrin, Ramesh Govindan, John Heidemann, "Impact of Network Density on Data Aggregation in Wireless Sensor Networks," *Proceedings of the International Conference on Distributed Computing Systems (ICDCS 2002)*, Vienna, Austria, July 2002.

[7] John Jubin and Janet D. Tornow, "The DARPA Packet Radio Network Protocols," *Proceedings of the IEEE*, 75(1):21-32, January 1987.

[8] Dongkyun Kim, C.-K. Toh, Yanghee Choi, "RODA : A new dynamic routing protocol using dual paths to support asymmetric links in mobile ad hoc networks," *Proceedings of the Ninth IEEE International Conference on Computer Communications and Networks (IEEE ICCCN 2000)*, Las Vegas, NV, USA, October 2000.

[9] Jinyang Li, Charles Blake, Douglas S. J. De Couto, Hu Imm Lee and Robert Morris, "Capacity of Ad Hoc Wireless Networks," *Proceedings of the Seventh ACM/IEEE International Conference on Mobile Computing and Networking*, Rome, Italy, July 2001.

[10] Sam Madden, Michael J. Franklin, Joseph Hellerstein, and Wei Hong, "TAG: a Tiny Aggregation Service for Ad-Hoc Sensor Networks," submitted for review, May 2002.

[11] D. Maltz and J. Broch, "Lessons from a Full-Scale Multihop Wireless Ad Hoc Network Testbed," *IEEE Personal Communications*, February 2001.

[12] Charles E. Perkins and Pravin Bhagwat, "Highly dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers," *Proceedings of the Conference on Communications Architectures, Protocols and Applications*, August 1994.

[13] Ranga Ramanujan, Sid Takkella, Jordan Bonney, Ken Thurber, "Source-Initiated Adaptive Routing Algorithm (SARA) for Autonomous Wireless Local Area Networks," *Proceedings of the 23rd IEEE Conference on Computer Networks*, October 1998.

[14] RF Monolithics, Inc., "ASH Transceiver TR1000 Data Sheet," available online at http://www.rfm.com/.

[15] Alec Woo and David Culler, "A Transmission Control Scheme for Media Access in Sensor Networks," *Proceedings of the Seventh ACM/IEEE International Conference on Mobile Computing and Networking*, Rome, Italy, July 2001.