Ad-Hoc Multicast Routing on Resource-Limited Sensor Nodes

Bor-rong Chen, Kiran-Kumar Muniswamy-Reddy, and Matt Welsh
Division of Engineering and Applied Sciences
Harvard University
Cambridge, MA 02138
{brchen,kiran,mdw}@eecs.harvard.edu

ABSTRACT

Many emerging sensor network applications involve mobile nodes with communication patterns requiring any-to-any routing topologies. We should be able to build upon the MANET work to implement these systems. However, translating these protocols into real implementations on resource-constrained sensor nodes raises a number of challenges. In this paper, we present the lessons learned from implementing one such protocol, Adaptive Demand-driven Multicast Routing (ADMR), on CC2420-based motes using the TinyOS operating system. ADMR was chosen because it supports multicast communication, a critical requirement for many pervasive and mobile applications. To our knowledge, ours is the first non-simulated implementation of ADMR. Through extensive measurement on *Motelab*, we present the performance of the implementation, TinyADMR, under a wide range of conditions. We highlight the real-world impact of path selection metrics, radio link asymmetry, protocol overhead, and limited routing table size.

Categories and Subject Descriptors:
C.2.2 [Computer-Communication Networks]: Network Protocols

General Terms: Experimentation, Measurement.

Keywords: Sensor Networks, Routing , Multicast.

1. INTRODUCTION

To date, much work on routing protocols in sensor networks has focused on forming stable routes to a single aggregation point. Many approaches have been proposed for spanning-tree formation, parent selection, and hop-by-hop data aggregation as data flows up the tree [27, 5, 16, 17]. These protocols are appropriate for networks consisting of stationary nodes that are primarily focused on data collection. However, several emerging applications for sensor networks require more general topologies as well as communication with mobile nodes. Examples include tracking firefighters in a burning building [28], data collection with mobile sensors [11, 12, 14], and monitoring the location and health status of disaster victims [15].

The mobile ad-hoc networking (MANET) community has developed a wide range of protocols for unicast and multicast routing using mobile wireless devices [19, 9, 10, 18]. Many of these protocols have been studied only under simulation using simplistic radio models, and do not consider issues such as bandwidth or memory limitations. In contrast, the sensor network community has demanded solutions that work on real hardware with limited resources. As a result, much of the MANET work has been overlooked by the sensor network community in favor of specially-tailored protocols that focus on energy management [22, 7], reliability [27, 26, 21], and in-network aggregation [16, 17].

Our goal is to bridge the gap between the mobile ad-hoc networking field and the state-of-the-art in sensor networks. By doing so, we hope to tap into the rich body of work in the MANET community, which may require reevaluating and redesigning these protocols as necessary. In particular, we identify several challenges to implementing MANET-based protocols on sensor nodes. The limited memory, computational power, and radio bandwidth deeply impact the implementation strategy. In addition, the realities of radio propagation, such as lossy and asymmetric links, require careful evaluation of path selection metrics.

This paper presents our experience with implementing a particular protocol, Adaptive Demand-Driven Multicast Routing(ADMR) [9], in TinyOS on MicaZ motes using the CC2420 radio. ADMR was chosen because it represents a fairly sophisticated and mature ad hoc multicast routing protocol, and was developed by an independent research group. To our knowledge, ADMR has never been implemented on real hardware, although its design has been well-studied in *ns-2* simulations assuming an 802.11 MAC. Our goal is to study the challenges involved in translating this style of protocol into a real implementation on resource-limited sensor nodes.

Several important lessons have emerged from this experience. The first is that communication performance is very sensitive to path selection metrics. The ADMR design attempts to minimize path hop count, but this can result in very lossy paths [4, 27]. We describe PATH-DR, a new metric that estimates the overall path delivery ratio using a simple hop-by-hop measurement of the CC2420's Link Quality Indicator (LQI). This metric differs from previous works in that it requires only a single packet reception and selects routes based on estimated delivery ratio across the entire path. The performance of PATH-DR is compared with conventional hop count and link quality metrics. The second lesson involves the impact of protocol overhead and practical limitations on data rates given the very limited radio bandwidth of IEEE 802.15.4. The third lesson deals with the impact of limited memory on routing protocol state. We evaluate several approaches for selectively dropping

REALMAN'06, May 26, 2006, Florence, Italy.

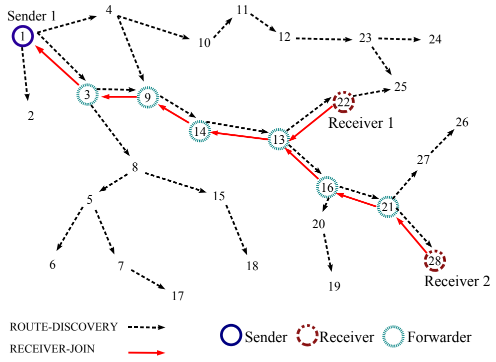

Figure 1: The ADMR route discovery process.

reverse-path information when the size of this state exceeds memory availability.

We present a detailed evaluation of our TinyOS implementation of ADMR running on Motelab [30], a 30-node indoor sensor network testbed. Motelab exhibits a great deal of variation in link quality and exercises the ADMR protocol in several ways. We present a comparison of several path selection policies, and demonstrate the impact of varying data rates, interference from other transmitters, and limitations on routing protocol state. Although length limits constrain us from presenting all of our results here; we refer the reader to [31] for full details.

2. ADMR BACKGROUND

Adaptive Demand-Driven Multicast Routing (ADMR) [9] is a multicast routing protocol designed for ad hoc networks in which nodes collaborate with each other to deliver packets. Data is multicast by sending packets to *group addresses* rather than individual *node addresses*. These packets will then be forwarded towards all the receivers belonging to a particular group along a forwarding tree established by the protocol. ADMR is fairly sophisticated and we elide many details from this discussion; we refer the reader to [9] for complete details.

2.1 Data forwarding

ADMR delivers packets from senders to receivers by routing each packet along a set of *forwarding trees* that are constructed on demand. Each tree is rooted at a single receiver and has leaves at each sender node for a group. ADMR's *route discovery* process assigns nodes in the network to be *forwarders* for a group based on measurements of potential routing paths between senders and receivers. Nodes assigned as forwarders rebroadcast data packets received for the corresponding group.

Forwarders are ignorant of the recipients for any group; rather, they simply rebroadcast group messages, and a single broadcast may be received by multiple nodes (including receivers or other forwarders). ADMR may also cause messages to traverse multiple routes from the sender to receiver. Each forwarder performs duplicate packet suppression by keeping track of the previously-transmitted sequence number for each $\langle sender, group \rangle$ pair; a forwarder will not rebroadcast the same data packet multiple times.

2.2 Route discovery

Route discovery is the process of assigning forwarders in the network and is crucial to ADMR. There are two ways of establishing forwarding states: *sender-initiated discovery* and *receiver-initiated discovery*. In sender-initiated discovery, senders initiate a network flood to find potential receivers. Receiver-initiated discovery reverses this process and has receivers flooding the network to discover senders. While we have implemented both discovery techniques, for brevity we only discuss sender-initiated discovery further.

The route discovery process (Figure 1) begins with senders sending out a ROUTE-DISCOVERY as a controlled network flood. Every node receiving this packet rebroadcasts the packet *once* allowing the message to propagate throughout the network. Upon receipt of a ROUTE-DISCOVERY, the node compares the hop count of the ROUTE-DISCOVERY to the lowest stored hop count (if any) from the sender generating the discovery.

If the new hop count is lower, the node stores three pieces of information: the *sender address* that originated the discovery, the *previous hop* from which the discovery message was received, and new *hop count* of the discovery message. This information is refreshed each time the sender initiates a new discovery process, as indicated by a sequence number in the message header. In this way, each node maintains the lowest hop count path from all sending nodes, as well as the previous hop from this sender.

When a receiver of the group receives ROUTE-DISCOVERY, it sends a RECEIVER-JOIN packet back to the original sender as a unicast message using path reversal. That is, the RECEIVER-JOIN is relayed along the lowest hop-count path back to the sender, using the stored previous hop information. Each intermediate node receiving a RECEIVER-JOIN configures itself as a *forwarder* for the corresponding $\langle sender, group \rangle$ pair. Once a sender receives any RECEIVER-JOIN, it can start broadcasting data packets for this group. The forwarding nodes will relay the messages until they reach the receivers.

2.3 Tree pruning

Tree pruning allows ADMR to deactivate unnecessary forwarders in the network. When a forwarder is no longer effective at delivering packets to downstream receivers, it should stop re-broadcasting messages to avoid wasting bandwidth. Likewise, if a receiver moves away or is no longer interested in the data from a certain group, there is no need to forward packets to that receiver. ADMR performs state expiration using *passive acknowledgments*. Whenever a forwarder rebroadcasts a packet, it listens for another downstream node to retransmit the packet that it just forwarded. If the packet is retransmitted by other nodes, then the forwarding state will remain valid. Otherwise, the forwarder will deactivate itself after an expiration period. Section 4.4 discusses the impact of passive acknowledgments versus active route reinforcement, which requires a receiver to periodically refresh forwarders with a RECEIVER-JOIN.

2.4 Routing state

To support the functions described above, ADMR maintains 3 tables on each node. *Node Table* stores the *previous hop* and *path cost* for each sender. It is indexed by $\langle sender, group \rangle$ pair. *Node Table* also stores the *sequence number* of the most recent message from each $\langle sender, group \rangle$ pair to suppress duplicate packets. *Membership Table* is also indexed by $\langle sender, group \rangle$ pair. It remembers whether a node is a receiver or forwarder for a given group address. *Sender Table* stores a list of group addresses for which a node is a sender.

3. IMPLEMENTATION

In this section we briefly describe *TinyADMR*, our implementation of the ADMR protocol for TinyOS-based mote platforms. TinyADMR is a complete reimplementation as faithful as possible based on details in [9] rather than an adapted version of the ns-2 implementation.

The most substantial change in TinyADMR is the departure from hop count as a path selection metric. As we discuss in Section 4.2, we have explored a range of metrics for picking good paths from senders to receivers. In TinyADMR, as ROUTE-DISCOVERYs are propagated through the network, the code provides a general notion of *path cost* that is stored in the Node Table.

TinyADMR is implemented as a NesC component which wires in several modules providing the protocol functionality. Most of the protocol functionality itself is implemented in a single module consisting of 1793 lines of commented NesC code. When compiled for the MicaZ mote, it requires 3544 bytes of ROM and 1563 bytes of RAM. Memory usage could be significantly reduced by removing debugging and instrumentation from the code.

3.1 Route discovery

Route discovery is implemented as described in Section 2.2, but TinyADMR allows use of various routing metrics other than hop count. To deal with topology change due to node mobility, the discovery process is invoked periodically; by default it is set to 15 sec, although for our experiments in Section 4 we decreased the interval to 5 sec to reduce the time to acquire measurements. All discovery and data packets are broadcast while RECEIVER-JOIN uses hop-by-hop acknowledgment and retransmission to ensure that it is routed to the sender. Each node along the path attempts to retransmit the RECEIVER-JOIN up to 5 times before dropping the message.

3.2 Routing metrics

The original ADMR protocol selects paths with the *minimum hop count*, which we call the MIN-HOP metric. As has been discussed elsewhere [4, 27], this choice of metric is not necessarily ideal, especially when link quality varies considerably. For example, MIN-HOP will prefer a short path over (potentially) very poor radio links rather than a longer path over high-quality links. MIN-HOP was appropriate in the original ADMR work which did not consider lossy radio links. However, in a realistic environment we expect it to have very poor performance. For this reason, We have investigated several alternate path selection metrics in our development of TinyADMR. For brevity we describe only two here. The comparison of these metrics are discussed in Section 4.2.

MAX-LQI. The first is to select the path with the "best worst link." In this metric the receiver selects the path with the highest *minimum* LQI value over all links in the path. Given a set of potential paths P and set of links L_p for each $p \in P$, we select the path $p^\star$:

$$p^\star = \arg\max_{p \in P} \min_{l \in L_p} \mathrm{LQI}(l)$$

We call this metric MAX-LQI. MAX-LQI attempts to find the path with the best "bottleneck," however, it does not have any way of differentiating between two paths with the same bottleneck link but different path characteristics.

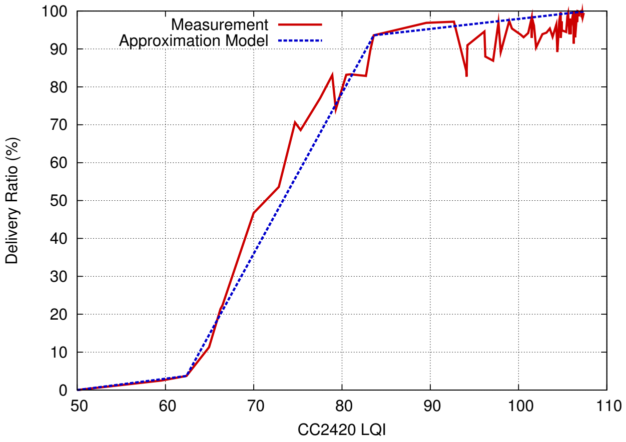

Figure 2: Relationship between LQI and delivery ratio in Motelab. ***Also shown is the piecewise linear model used to map LQI observations to LDR estimates.***

PATH-DR. Because our goal in ADMR is to maximize the overall *path delivery ratio* (PDR), directly using the path with the best PDR would be ideal. However, directly measuring PDR requires measuring the hop-by-hop LDR along the path. This requires multiple rounds of message exchange between neighboring nodes, incurring additional messaging overhead. This is the approach used by ETX [4] and MintRoute [27].

Rather than directly measure LDR, we have found that there is a high correlation between the LQI and LDR observed on each link. Figure 2 plots the pairwise LQI and LDR over an extensive set of measurements on Motelab. From this data we can derive a simple *model* mapping LQI to LDR; a piecewise linear approximation appears to work well for this data set. Because LQI can be observed from a *single packet reception*, using this information to predict the previous-hop LDR allows us to produce the PATH-DR metric, which selects the path $p^\star$ such that:

$$p^\star = \arg\max_{p \in P} \prod_{l \in L_p} \mathrm{ESTLDR}(l)$$

where $\mathrm{ESTLDR}(l)$ is the estimated LDR of the link from the LQI of the received discovery message, derived from our empirical model shown in Figure 2.

3.3 Route pruning

To perform route pruning, each Node Table entry is assigned a lifetime when it is created. The lifetime of each entry is refreshed based on a *path reinforcement policy*. In TinyADMR, two path reinforcement policies are implemented: *active reinforcement* and *passive reinforcement*. Active reinforcement refreshes a forwarder membership whenever a new RECEIVER-JOIN is received. Passive reinforcement refreshes the forwarder membership based on passive acknowledgment of each transmitted data packet. Details and impact of these two reinforcement methods are discussed in Section 4.4.

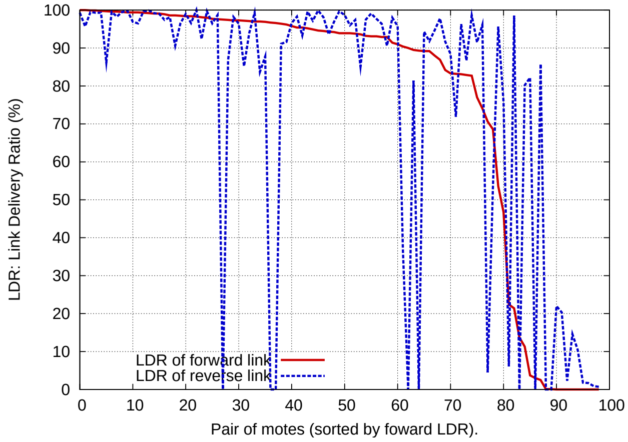

Figure 3: Link delivery ratio (LDR) asymmetry observed in our testbed.

3.4 Routing state

As specified in Section 2.4, each node maintains three tables in order to support the multicast functionality. The size of the Node Table depends on how many nodes are in the network that are acting as multicast senders and receivers. The size of the Sender and Membership Tables can be determined by the number of multicast groups expected to exist in the network. Having enough space in the routing tables, especially the Node Table, is critical because ADMR will not perform properly without large enough table sizes.

4. EVALUATION AND LESSONS LEARNED

Implementing ADMR in TinyOS and obtaining good communication performance in a realistic network environment has not been a trivial undertaking. There is a significant disconnect between the original ADMR protocol as published and the conditions encountered in a real sensor network. In particular, ADMR assumes symmetric links, uses hop count as its path selection metric, and ignores memory space issues when maintaining routing tables. This section presents a detailed evaluation of our TinyOS-based ADMR implementation and a series of lessons learned in the process of developing and tuning the protocol. We believe these lessons will be useful to other protocol designers working with resource limited sensor nodes.

4.1 Evaluation environment

We have focused exclusively on real implementation and evaluation on a sensor node testbed, rather than simulations, to understand the performance and behavior of TinyADMR. All of our results have been gathered on Motelab [30][1], an indoor testbed of 30 MicaZ motes installed over three floors of our Computer Science building. This testbed provides facilities for remote reprogramming of each node over an Ethernet back-channel board (the Crossbow MIB600). Each node's serial port is exposed through a TCP port permitting detailed instrumentation and debugging. Motes are installed in various offices and labs and are typically placed on shelves at a height of 1-2 m.

Because of the relatively sparse node placement, this testbed exhibits a high degree of variation in radio link quality and many asymmetric links. Figure 3 shows the forward and reverse *link delivery ratio* (LDR) calculated for every pair of nodes in the testbed. Using a technique similar to the SCALE benchmark [3], the link delivery ratio is measured by having each node broadcast a fixed number of messages in turn, while all other nodes record the number of messages received from each transmitter. The LDR is the ratio of received messages to transmitted messages for each pair of nodes. As the figure shows, the LDR is highly variable and often asymmetric.

The CC2420 radio provides an internal Link Quality Indicator (LQI) for each received message [1]. This value represents the ability of the CC2420 to correlate the first eight 32-chip symbols following the start-of-frame delimiter, and has an effective range from 110 (highest quality) to 50 (lowest quality). As we will discuss in Section 3.2, the LQI is highly correlated with the link delivery ratio and is useful for route discovery.

In each of the cases presented below, we measure routes between 28 different sender/receiver pairs. These pairs were selected to present a diverse view of potential routes in our testbed. Four of the 28 pairs were within a single radio hop, 11 were within two radio hops, and the remaining 13 pairs were chosen so that the endpoints were on opposite ends of the building. In each case, senders generated data at a rate of 5 Hz, each experiment was run for 100 sec, and route discovery process was invoked every 5 sec. Once the benchmark is started, we wait for 30 seconds before collecting packet reception statistics, to avoid measuring warmup effects.

Original ADMR evaluation. It is worth contrasting our environment to that used in the original ADMR paper. In [9], ADMR was measured using *ns-2* simulations with a network of 50 nodes roaming in a 1500 m × 300 m area. A 2 Mbps 802.11 radio with a radio range of 250 m was simulated; this implies that most nodes are within a small number of hops of each other. Most importantly, nodes have perfect connectivity to all other nodes within this range and links are always symmetric.

4.2 Impact of path selection metrics

This section describes the results and comparison of the three routing metrics described in Section 3.2.

Figure 4 shows a CDF of the path delivery ratio for the set of 28 paths. As the figure shows, PATH-DR results in the highest delivery ratio over all paths, with nearly all paths resulting in a PDR of over 80%, and a median of 92.4%. MAX-LQI also performs well, with a median PDR of 79.6%. MIN-HOP is noticeably worse, with a median PDR of just 60.7%.

PATH-DR and MAX-LQI achieve higher robustness at the cost of higher overhead. With all three metrics, multiple routing paths may be active between a sender and receiver at once depending on timeout of the path reinforcement policy. The existence of multipath routes should result in higher path delivery ratios. Figure 5 shows the overhead for each path selection metric in terms of the ratio between the total number of transmitted messages and the number of messages generated by each sender. This ratio is at least as high as the number of routing hops from sender to receiver, and will be higher when multiple routes are involved.

The path length for each RECEIVER-JOIN is shown in Figure 6, while the total number of active forwarders for each node pair is shown in Figure 7. The median path length for MIN-HOP is 2 hops, where it is about 4 hops for both PATH-DR and MAX-LQI. The

---

[1] http://motelab.eecs.harvard.edu/

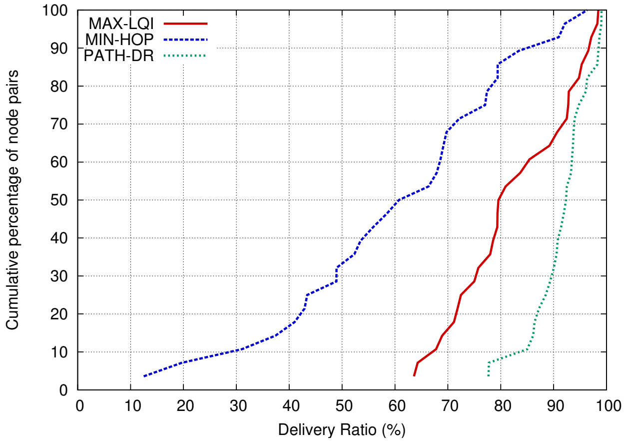

Figure 4: Comparison of MIN-HOP, MAX-LQI, and PATH-DR routing metrics. ***This CDF shows the path delivery ratio measured over 28 separate pairs of nodes using each of the three metrics. PATH-DR produces the best results with 50% of the paths obtaining a delivery ratio of over 92.4%.***

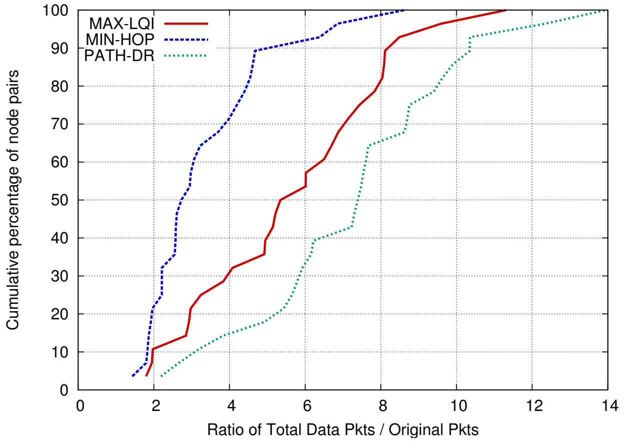

Figure 5: Overhead incurred by each path selection metric. ***This CDF shows the ratio of the number of data packets transmitted in the network (including forwarded messages) to the number of data packets originated by each sender. For example, for 70% of the 28 paths, MAX-LQI resulted in an overhead of 6 transmissions for every original packet.***

number of forwarders is somewhat higher than the number of hops because PATH-DR and MAX-LQI may activate multiple paths on subsequent route selection phases. The impact of pruning these extra routes is presented in Section 4.4.

These results show that the routing selection metric has a large impact on performance and communication overhead. The PATH-DR metric provides high reliability using a simple model mapping the CC2420's LQI to LDR, making it straightforward to implement without incurring additional measurement overhead. We can imagine a wide range of alternate path selection metrics as part of future work.

4.3 Impact of limited bandwidth

Another shortcoming of most MANET protocol designs is that they assume relatively high link bandwidths. The original ADMR protocol was evaluated using a simulated 802.11 network with a raw transmission rate of 2 Mbps. MAC overhead leaves roughly 1 Mbps to applications. In contrast, 802.15.4-based radios provide substantially less capacity. While 802.15.4 operates at a nominal transmission rate of 250 Kbps, our measurements of the CC2420 using the default radio stack in TinyOS results in an application data rate of just 25 Kbps with small packets (28 bytes) and up to 60 Kbps for larger packets (100 bytes). This is 17 times less than 802.11 at 2 Mbps (1 Mbps for applications), or 93 times less than 802.11b operating at 11 Mbps (5.5 Mbps for applications).

For these reasons we expect protocol overheads to have a serious impact on the performance of ADMR on 802.15.4. We have not attempted to minimize these overheads; rather, our goal is to demonstrate the practical implications of limited channel bandwidth.

Protocol overhead in ADMR arises primarily due to route discovery and route reply messages. In sender-initiated discovery, each sender periodically floods the network, which allows node tables to be maintained on each intermediate node. When a receiver wishes to establish a path it sends a route reply back to each sender. The periodic per-sender floods induce the highest overhead in ADMR and unfortunately scale with network size. For example, in a network of 30 nodes propagating per-node floods every 5 sec, the per-node protocol overhead is $30^2/5 = 180$ packets/sec.

Because the size of our testbed is limited, we cannot generate an arbitrary amount of protocol overhead (which might be seen in a much larger network) directly. Instead, we cause all nodes in the network to generate *interference* packets at a rate that we control. We then show the impact on the achieved delivery ratio for several paths as this interference rate varies.

Figure 8 shows the results of this experiment with the per interfering node interference rate increasing from 0 to 50 packets/sec. To eliminate effects where a node drops its own transmissions because it is also generating interference messages, whenever a node is configured as a forwarder, it generates no interference messages of its own. While this is not entirely realistic (a node will still propagate discovery floods while it is acting as a forwarder), we wanted to avoid losing data transmissions due to queue overflow on the transmitter.

As the figure shows, the path delivery ratio drops rapidly with even a modest amount of interference traffic. Keep in mind that this is *not* because forwarders are dropping outgoing packets (due to MAC backoff or queue overflow). The only explanation is that nodes are unable to *receive* packets as well in the presence of interfering traffic. That is, the interference traffic "jams" receivers along the ADMR path and prevents them from correctly decoding incoming messages. This is likely due to collisions caused by hidden terminal and capture effects.

4.4 Impact of route pruning

In ADMR, nodes are configured as routers when they receive a route reply message from a receiver. Over time, different routes may be activated as link conditions change. Also, at any given time, multiple routes may exist between a single pair of nodes. By pruning redundant routes from the network over time, communication overheads can be reduced, although this may also have a negative impact on path reliability.

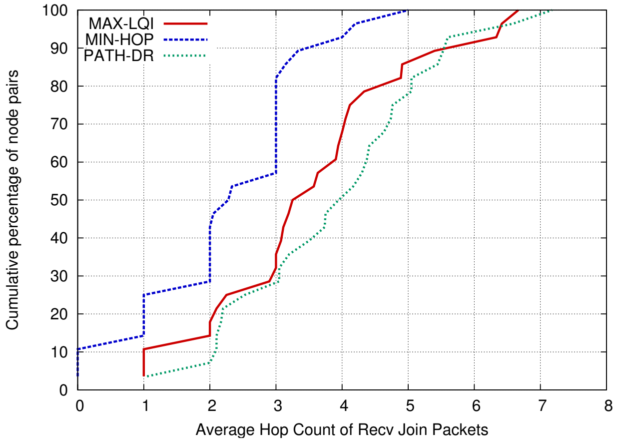

Figure 6: Path lengths for RECEIVER-JOIN for each path selection metric. ***This CDF shows the length of the paths selected by receivers for each metric. The MIN-HOP metric minimizes the hop count, while MAX-LQI and PATH-DR incur some path stretch because they focus on higher-quality links.***

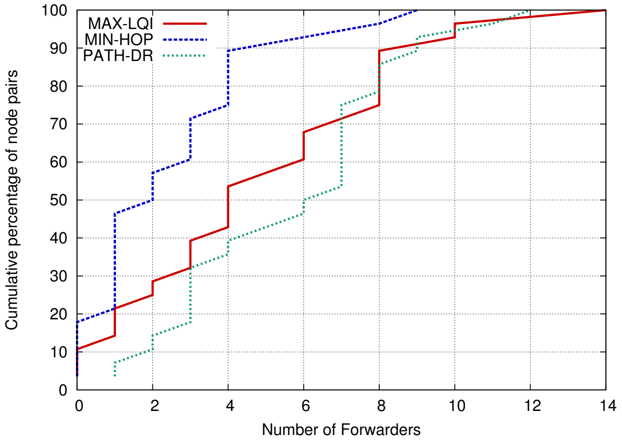

Figure 7: Number of active forwarders for each path selection metric. ***This CDF shows the number of forwarders that are active while routing data for each of the 28 paths. The number of forwarders is not identical to the path length in Figure 6 because multiple paths may be active.***

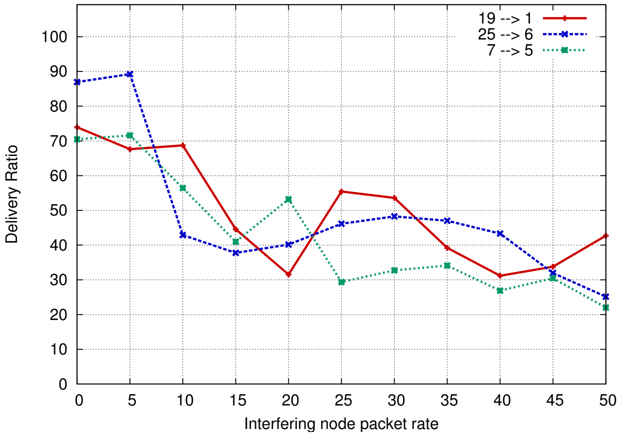

Figure 8: Delivery ratio for three representative paths as the amount of interference traffic is increased.

We investigate the impact of two approaches to path pruning in ADMR. The first, *active reinforcement*, requires that nodes continue to receive route reply messages from a receiver in order to stay active as forwarders. If a node has not received a route reply for 10 sec, it clears its forwarder status. This time is equivalent to 2 discovery cycles. The second approach, *passive reinforcement*, causes a node to remain active as a forwarder as long as it overhears another node retransmitting its own messages (or it continues to receive route replies).

Figure 9 compares the delivery ratio for the active and passive reinforcement schemes. Not surprisingly, passive reinforcement keeps more forwarders active, resulting in much higher reliability than active reinforcement.

4.5 Impact of limited node state

The final lesson that we explore involves the impact of limited node state. Sensor nodes such as the Telos and MicaZ have notoriously small memory sizes: the MSP430 microprocessor has only 10 KB of RAM, while the Atmega 128L has just 4 KB. We cannot expect that the routing layer can consume an arbitrary amount of memory to store its routing state. This is especially true if there is a substantial application running on top of the routing layer that has its own memory requirements.

The node table maintained by every ADMR node potentially contains one entry for every other node in the network. In our implementation, each entry consumes 9 bytes. In a large network, it is clear that the number of entries in this table can quickly saturate memory.

The original ADMR paper [9] suggests using an LRU strategy to prune node table entries over time. However, in a network with many active senders it may not be possible to guarantee an upper bound on the node table size. We are concerned with how to deal with *overflow* in the node table given some fixed limit on its size.

Upon receipt of a discovery message, ADMR will consult the node table and either update the existing entry for this sender, or attempt to create a new entry. If the table is full, we must either drop the new entry or evict some other entry to make room. Dropping a node table entry has two effects. The first is that the node loses information on the routing cost from the sender. This is only needed when establishing new routes, so these entries are only needed shortly after a discovery message has been received (in case the receiver wishes to reinforce this route).

The second effect is that the last sequence number received from this node, used for duplicate suppression, is lost. This information is required for all senders for which this node is a forwarder. This suggests that ADMR should keep the last sequence number and reverse-path information in separate tables, although this is not the case in the current protocol design.

We explore several different policies for evicting table entries. The most naive policy simply drops the new entry if the table is full, only allowing updates to entries already in the table. Another simple policy is FIFO, which drops the oldest entry (where entries are ordered by time of insertion). FIFO is intended to time out

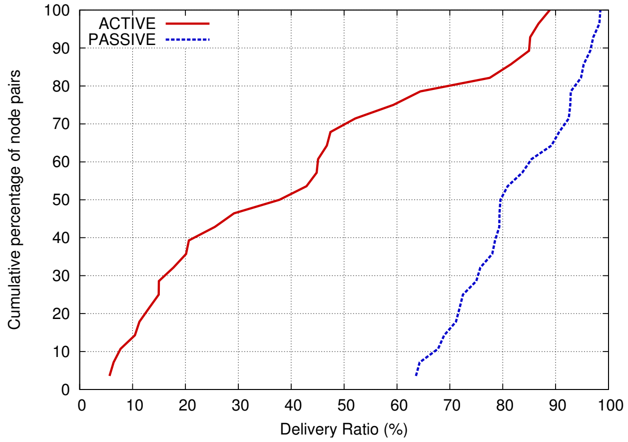

Figure 9: Comparison of delivery ratio for active vs. passive route reinforcement.

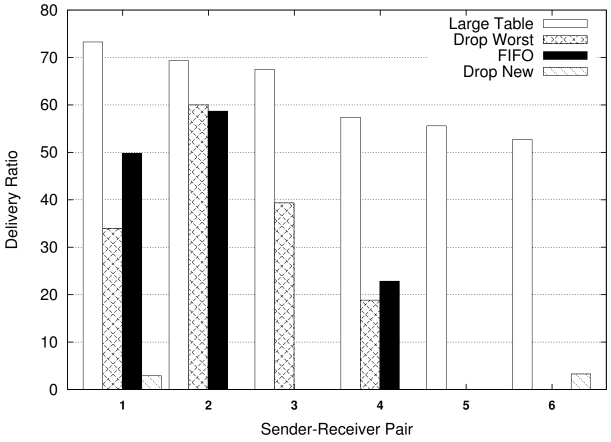

Figure 10: Comparison of delivery performance with different node table management strategies. ***This experiment uses 6 senders and 1 receiver. In each case except for* Large Table*, the node table size is limited to 4 entries to simulate the effect of scaling in a large network. None of the proposed schemes for prioritizing node table entries works well in all cases.***

stale entries from the table in favor of new entries. However, if the routing cost for the new entry is very high, it may not be worthwhile maintaining information on this route.

A better approach may be to maintain node table entries for high-quality routes, with the expectation that this node will be called upon to act as a forwarder. In some sense, the impact of dropping discovery messages for low-quality paths should not be too severe.

Figure 10 shows a comparison of each of these table-management policies with 6 senders and 1 receiver. We emulate the impact of a large network by artificially limiting the node table size to 4 entries on each node. This implies that nodes will not be able to maintain routing state for all 6 senders. As the results show, none of the proposed policies works well in all cases. The naive drop-new-entry policy performs very poorly. The FIFO and drop-worst policies work reasonably for only for 3 out of the 6 routes.

Another approach to managing limited memory size is to swap node table entries to external memory (such as flash) on demand. We have not yet explored this approach, although the results presented above suggest it may be necessary to do so in large networks.

5. RELATED WORK

5.1 MANET routing protocols

Researchers in the MANET community have put in a great deal of effort into the design of routing protocols for ad hoc networks. For unicast use, DSDV [18], AODV [19] and DSR [10] are three of the most commonly-studied routing protocols for MANET applications. We do not describe these protocols in detail but they can be divided into two categories: Proactive and On-demand. Proactive protocols, such as DSDV, periodically exchange routing table among neighboring nodes while on-demand protocols, such as AODV and DSR, only acquire routes when needed.

A wide range of ad hoc multicast routing protocols have also been proposed. For brevity we only discuss three examples here: ADMR [9], ODMRP [13] and MAODV [20]. ODMRP and MAODV are both similar to ADMR. The route establishment mechanism is basically equivalent in all three protocols, although ODMRP only supports sender-initiated discovery model. The major differences between ODMRP and ADMR are in the tree-pruning mechanisms and how broken links are repaired.

In MAODV, route discovery is also carried out with network flooding. However, MAODV does not differentiate data senders from receivers in the multicast tree. All nodes in a multicast group answer the discovery message with a unicast packet. The node that sends out the discovery message picks the best route according to the multiple reply messages. MAODV also broadcasts a *hello* message periodically to detect link failure.

5.2 Studies of real-world link quality

Studies of link-level data delivery characteristics of wireless networks have confirmed that the assumption of fixed-range, symmetric links is unrealistic [2, 3, 6]. New routing metrics based on dynamic link quality estimation [4, 27] have been proposed to alleviate this problem by selecting longer but more reliable routes. Unfortunately, the MANET routing protocols described in this section all adopt hop count as the path routing cost. Therefore, it is unclear from the current literature how well the proposed MANET multicast routing protocols will work if they are deployed in a wireless network.

5.3 Routing in TinyOS-based sensor networks

Routing in sensor networks has primarily been focused on building a single spanning tree that routes messages from every node to a single base station [27, 5, 16, 17, 8]. Such a global spanning tree is useful for a broad range of sensor network applications involving network-wide data collection [24, 25, 23].

Woo et al. [27] carefully studied design strategies for many-to-one spanning trees in sensor networks. They found maintaining a spanning tree with highly-reliable links is non-trivial and requires dynamic link estimation on each sensor node. The evaluation is based on a simulation model derived from two-node link quality measurements. Yarvis et al. [29] created and evaluated an implementation of DSDV on motes. They also point out that high-quality link outperforms shortest-hop-count paths.

6. FUTURE WORK AND CONCLUSIONS

This paper has presented an investigation of the issues that arise when translating MANET-based protocol designs into a sensor network context. As sensor networks become more widespread, new applications will be developed that present a broad set of communication requirements. Given that the MANET community has invested a great deal of effort into routing protocols for mobile wireless environments, we believe that there is real value in understanding to what extent this work can be reapplied.

Many of the lessons arising from our TinyOS-based implementation of ADMR stem from the enormous differences in the assumed hardware environment. ADMR was developed for relatively high-powered devices with significantly more radio bandwidth and memory than is found on typical motes. It is not surprising, then, that we faced some challenges implementing ADMR on this platform. We feel that it is significant that we were able to produce a working and fairly robust implementation of ADMR despite these challenges.

While our goal was to remain faithful to the original ADMR design as much as possible, the most substantial modification was the introduction of alternate path-selection metrics. Minimizing hop count performs poorly, while a simple LQI-based estimation of the path delivery ratio works well and incurs no additional measurement overhead. We have also investigated techniques for active versus passive route reinforcement as well as managing limited routing state.

Our results demonstrate that there is still significant work to be done if ADMR is to be effective in large networks. With a large number of nodes, protocol overhead will readily saturate available bandwidth. Most MANET protocols use a fixed transmission rate for protocol packets such as discovery messages. Dynamically tuning these rates based on interference traffic or node density would permit overhead to scale with available bandwidth.

Protocol state management under severe memory limitations is another area for future work. We have explored various node table eviction policies, although results demonstrate that deliberately dropping this state has an adverse effect on performance. More efficient data structure design and swapping state to external flash may be viable solutions.

7. REFERENCES

[1] CC2420 Product Information. `http://www.chipcon.com/index.cfm?kat_id=2&subkat_id=12&dok_id=115`.

[2] D. Aguayo, J. Bicket, S. Biswas, G. Judd, and R. Morris. Link-level measurements from an 802.11b mesh network. *SIGCOMM Comput. Commun. Rev.*, 34(4):121–132, 2004.

[3] A. Cerpa, N. Busek, and D. Estrin. SCALE: a tool for Simple Connectivity Assessment in Lossy Environments. Technical report, September 2003.

[4] D. S. J. D. Couto, D. Aguayo, J. Bicket, and R. Morris. A high-throughput path metric for multi-hop wireless routing. In *MobiCom '03: Proceedings of the 9th annual international conference on Mobile computing and networking*, pages 134–146. ACM Press, 2003.

[5] D. Ganesan, D. Estrin, and J. Heidemann. Dimensions: why do we need a new data handling architecture for sensor networks? *SIGCOMM Comput. Commun. Rev.*, 33(1):143–148, 2003.

[6] 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, Feburary 2002.

[7] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *Proc. the 33rd Hawaii International Conference on System Sciences (HICSS)*, January 2000.

[8] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *Proc. International Conference on Mobile Computing and Networking*, Aug. 2000.

[9] J. G. Jetcheva and D. B. Johnson. Adaptive Demand-Driven Multicast Routing in Multi-Hop Wireless Ad Hoc Networks. In *2001 ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc2001)*, pages 33–44, October 2001.

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

[11] W. Kaiser, G. Pottie, M. Srivastava, G. Sukhatme, J. Villasenor, and D. Estrin. Networked Infomechanical Systems (NIMS) for Ambient Intelligence. Invited contribution to Ambient Intelligence, Springer-Verlag, 2004.

[12] A. Kansal, A. A. Somasundara, D. D. Jea, M. B. Srivastava, and D. Estrin. Intelligent fluid infrastructure for embedded networks. In *MobiSYS '04: Proceedings of the 2nd international conference on Mobile systems, applications, and services*, pages 111–124, New York, NY, USA, 2004. ACM Press.

[13] S.-J. Lee, M. Gerla, and C.-C. Chiang. On-demand multicast routing protocol. In *IEEE Wireless Communications and Networking Conference, WCNC '99*, pages 1298–1304, September 1999.

[14] T. Liu, C. M. Sadler, P. Zhang, and M. Martonosi. Implementing software on resource-constrained mobile sensors: experiences with impala and zebranet. In *MobiSYS '04: Proceedings of the 2nd international conference on Mobile systems, applications, and services*, pages 256–269, New York, NY, USA, 2004. ACM Press.

[15] K. Lorincz, D. Malan, T. R. F. Fulford-Jones, A. Nawoj, A. Clavel, V. Shnayder, G. Mainland, S. Moulton, and M. Welsh. Sensor Networks for Emergency Response: Challenges and Opportunities. *IEEE Pervasive Computing*, Oct-Dec 2004.

[16] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. TAG: A Tiny AGgregation Service for Ad-Hoc Sensor Networks. In *Proc. the 5th OSDI*, December 2002.

[17] S. Nath, P. B. Gibbons, S. Seshan, and Z. R. Anderson. Synopsis diffusion for robust aggregation in sensor networks. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 250–262, New York, NY, USA, 2004. ACM Press.

[18] C. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. In *ACM SIGCOMM'94 Conference on Communications Architectures, Protocols and Applications*, pages 234–244, 1994.

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

[20] E. M. Royer and C. E. Perkins. Multicast operation of the ad-hoc on-demand distance vector routing protocol. In *Fifth Annual ACM/IEEE International Conference on Mobile Computing and Networking, Mobicom '99*, pages 207–218, August 1999.

[21] Y. Sankarasubramaniam, O. B. Akan, and I. F. Akyildiz. ESRT: Event-to-Sink Reliable Transport in Wireless Sensor Networks. In *Proceedings of ACM MobiHoc'03*, pages 177–188, Annapolis, Maryland, USA, June 2003.

[22] C. Schurgers, V. Tsiatsis, S. Ganeriwal, and M. Srivastava. Topology management for sensor networks: exploiting latency and density. In *MobiHoc '02: Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing*, pages 135–145, New York, NY, USA, 2002. ACM Press.

[23] G. Simon, M. Maroti, A. Ledeczi, G. Balogh, B. Kusy, A. Nadas, G. Pap, J. Sallai, and K. Frampton. Sensor network-based countersniper system. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 1–12, New York, NY, USA, 2004. ACM Press.

[24] R. Szewczyk, A. Mainwaring, J. Polastre, and D. Culler. An analysis of a large scale habitat monitoring application. In *Proc. the Second ACM Conference on Embedded Networked Sensor Networks (Sensys)*, November 2004.

[25] R. Szewczyk, E. Osterweil, J. Polastre, M. Hamilton, A. Mainwaring, and D. Estrin. Habitat monitoring with sensor networks. *Communications of the ACM, Special Issue: Wireless sensor networks*, 47(6):34–40, June 2004.

[26] C.-Y. Wan, A. T. Campbell, and L. Krishnamurthy. Psfq: a reliable transport protocol for wireless sensor networks. In *WSNA*, pages 1–11, 2002.

[27] A. Woo, T. Tong, and D. Culler. Taming the underlying challenges of reliable multihop routing in sensor networks. In *Proc. the First ACM Conference on Embedded Networked Sensor Systems (SenSys 2003)*, November 2003.

[28] P. Wright et al. Fire information & rescue equipment. `http://fire.me.berkeley.edu/indexmain.php?main=home.php&title=FIRE\%20P%roject:\%20Overview`.

[29] M. Yarvis, W. S. Conner, L. Krishnamurthy, J. Chhabra, B. Elliott, and A. Mainwaring. Real-world experiences with an interactive ad hoc sensor network. In *Proceedings of the International Workshop on Ad Hoc Networking*, August 2002.

[30] Geoff Werner-Allen, Pat Swieskowski, and Matt Welsh. MoteLab: A Wireless Sensor Network Testbed. In *Proceedings of the Fourth International Conference on Information Processing in Sensor Networks (IPSN'05), Special Track on Platform Tools and Design Methods for Network Embedded Sensors (SPOTS)*, April 2005.

[31] Bor-rong Chen, Kiran-Kumar Muniswamy-Reddy, and Matt Welsh. Lessons Learned from Implementing Ad-Hoc Multicast Routing in Sensor Networks. *Harvard University Technical Report TR-22-05*, November 2005.