Magnet: Robust and Efficient Collection through Control and Data Plane Integration

Paper # 1569122052: 14 pages

Abstract

Despite being a core networking primitive, collection protocols today often suffer from poor reliability (e.g., 70%) in practice, and heavily used protocols have never been evaluated in terms of communication efficiency. Using detailed experimental studies, we describe three challenges that cause existing collection protocols to have poor reliability and waste energy: inaccuracies in link estimation, link dynamics, and transient loops.

In this paper we present Magnet, a robust, efficient, and hardware-independent collection protocol. Magnet uses three novel techniques to address these challenges. Magnet's link estimator addresses the inaccuracies in link estimation by using feedback from both the data and control planes, using information from multiple layers through narrow, platform-independent interfaces. Second, Magnet addresses link dynamics by using the Trickle algorithm for control traffic, sending few beacons in stable topologies yet quickly adapting to changes. Finally, Magnet addresses transient loops by using data traffic as active topology probes, quickly discovering and fixing routing failures.

Magnet runs on six different mote platforms and we have tested it on four testbeds. In most experiments, Magnet achieves 99% reliability, and in some cases 99.9%. In the most challenging testbed, the state-of-the-art collection protocol today (MultiHopLQI) achieves 70% reliability: Magnet achieves 97%. Magnet achieves this ten-fold reduction in dropped packets with 25% fewer transmissions. Magnet works seamlessly on top of existing low-power MAC layers. Together, these results suggest that Magnet can be the robust, efficient collection layer that so many sensor network applications and protocols need.

1 Introduction

Collection trees are a core building block for sensor network applications and protocols. In their simplest use, collection trees provide the unreliable, datagram routing layer that most deployments use to gather data [26, 39, 43]. Additionally, tree routing forms the underlying topology of most higher level routing [12, 14, 27] and transport [20, 28, 29, 33] protocols.

But despite its key role in so many systems, more often than not we see collection exhibit persistent problems. Real deployments have low delivery ratios of anywhere from 2-68% [21, 26, 39, 43]. Although the common routing metric is expected transmissions (ETX) [8], to the best of our knowledge there are no published evaluations of how efficient protocols actually are: it may be that they are sending twice as many packets as they need to.

One common approach to deal with these issues has been to integrate the protocol closer to higher or lower layers. In the case of integrating the collection tree more closely to the higher layers, MintRoute [46] requires that an estimate of the rate of data traffic be hardcoded in the program, and both its link estimation and control traffic depend on such fixed parameter. In the case of integrating the collection tree more closely to the lower hardware layer, MultiHopLQI [37] takes advantage of the functionality only a single radio chip, the CC2420, provides. Despite all of this vertical integration, however, collection protocols continue to perform poorly.

Through extensive measurement studies, we find that three phenomena cause most packet losses in collection protocols. First, links are highly dynamic, swinging between high and low delivery rates on the time scale of hundreds of milliseconds. This causes link estimation from periodic beacons to be inherently inaccurate. Second, physical layer information (such as RSSI and the CC2420's LQI) suffers from *sampling bias*. A protocol only measures them on received packets, so highly dynamic links can report high values yet lose many packets. Third, dynamic topologies inevitably form (possibly transient) loops, and protocols drop looped packets. This is the case even in path-vector protocols like BGP, designed to avoid loops altogether [30]. While dropping these packets is not a problem for 80% or even 90% reliability, loops are common enough that this policy precludes more than one nine of reliability.

In this paper we present Magnet, a robust, efficient, and hardware-independent collection protocol. Magnet incorporates three novel mechanisms to address these challenges. First, it uses a hybrid link estimator that incorporates active feedback from the data path, enabling Magnet to be highly agile and respond to broken links within a few packets. Second, it uses an adaptive control traffic rate based on the Trickle algorirthm [23]. As long as the routing gradient in the network is consistent, nodes slowly send fewer and fewer control packets. When the network detects a loop or other inconsistency, nodes send control traffic more quickly to repair the topology. Finally, Magnet uses data traffic to actively probe the topology, detecting routing problems and repairing them as needed. The effectiveness of these three techniques suggests that the failures of many protocols today may be addressed by a judicious increase in the integration between the data and control planes, indicating a way in which wireless protocols are fundamentally different than their wired siblings.

We have tested Magnet on four platforms, and evaluated it by running extensive experiments in four testbeds, comparing it against the state of the art collection protocol today, MultiHopLQI. In one testbed we measured, Magnet drops one tenth the data (97% vs 70% delivery ratio) while sending 29% fewer packets. In more forgiving environments, we have observed 99.9% delivery ratios

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

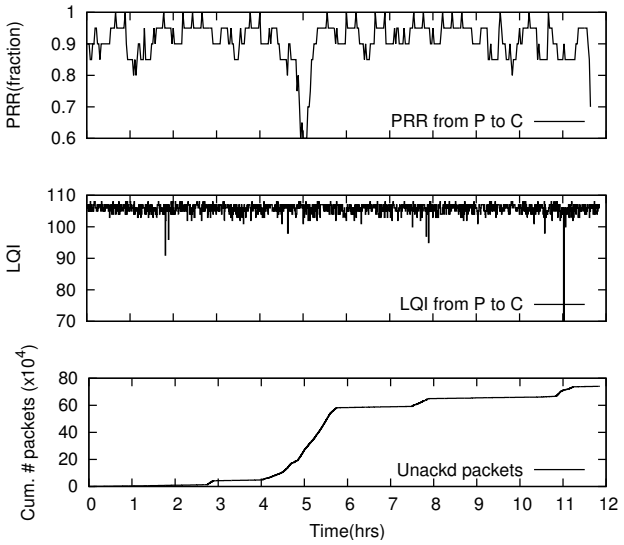


Figure 1. Unaware of the reduced PRR, MultiHopLQI attempts to deliver packets on the same link between the forth and sixth hour causing increased number of retransmissions due to unacknowledged packets.

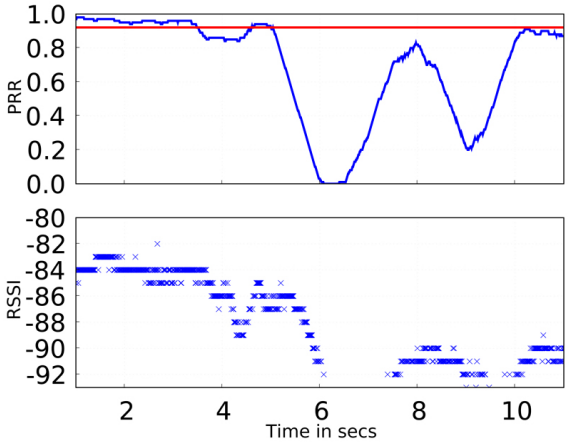


Figure 2. RSSI and other physical-layer measurements can vary significantly over short time periods. This causes measurement bias, where received packets have a high RSSI, yet periods of low RSSI cause packet losses. The red line shows the long term average PRR.

over 5 hops, with the 0.1% packet losses being due to receiving link-layer acks for packets that do not arrive at the network layer. Unlike MultiHopLQI, Magnet does not depend on specific networking hardware: we have run it on CC1000, CC2420, CC1100, and CC2500-based nodes. The protocol itself is 5.5kB of code on an MSP430, and in its default configuration it requires approximately 1kB of RAM, most of which is its packet forwarding queue. Others have also reported running Magnet on two additional platforms.

While Magnet strives to provide high reliability, it does not provide the 100% reliability of an end-to-end transport protocol for a number of reasons. First, since Magnet does not provide reverse routes from the collection points to a specific node, adding these would require additional topology information and maintenance by the nodes. Second, not all applications require 100% reliability, and incorporating these mechanisms would impose the costs and complexity on all clients. Lastly, it is still an open question whether a single end-to-end transport protocol could be made to satisfy all sensornet applications, and Magnet provides a solid base on top of which different alternatives can be implemented and compared.

Most of Magnet's design decisions come from experiences addressing the very different topologies and behaviors the different environments introduce. In this paper, we focus on Tutornet, which is the most dynamic of the four environments and presented the greatest challenges to robust, efficient collection. In Section 2, we examine why testbeds like Tutornet are so challenging and how that causes existing protocols, such as MultihopLQI, to fail badly. Based on these observations, we present Magnet's high-level design in Section 3, and its hybrid link estimator in Section 4. Section 7 presents a series of detailed experimental studies of Magnet and MultihopLQI in Tutornet, as well as brief results from the Mirage testbed at Intel Research Berkeley. Section 8 presents prior work that guided Magnet's design, and Section 9 concludes.

2 Motivation

Implementing robust and efficient wireless protocols is notoriously difficult. Collection tree protocols are no exception. At first glance, collection protocols are very simple. They provide best-effort, unreliable packet delivery to one of the data sinks in the network. They are typically anycast, and simply deliver packets to the closest or best sink. Having a robust, highly reliable, and efficient collection protocol benefits almost every sensor network application today, as well as the many transport, routing, overlay, and application protocols that sit on top of collection trees.

But despite providing a simple service that is fundamental to so mant systems, and being in use for almost a decade, collection protocols today typically suffer from poor performance. Deployments observe delivery ratios of anywhere from 2-68% [21, 26, 39, 43]. Furthermore, although collection protocols are intended to be energy efficient and minimize transmissions, to the best of our knowledge there are no published evaluations of their efficiency.

Furthermore, it is not clear *why* collection protocols perform well in controlled situations yet poorly in practice, even at low data rates. To better understand the causes of these failures, we ran a series of experiments on a number of public testbeds, including Mirage at Intel Research Berkeley, Tutornet at USC, Motelab at Harvard, and a private testbed in a research lab. We found that three phenomena were the dominant causes: link dynamics, packet sampling bias, and transient loops.

2.1 Link Dynamics

Protocols today use periodic beacons to maintain their topology and estimate link qualities. The beaconing rate introduces a tradeoff between agility and efficiency: a faster rate leads to a more agile network but higher cost, while a lower rate leads to a slower-to-adapt network and lower cost. The inherent assumption in these measurement approaches is that beacon packets accurately measure the communication quality that data packets see. Early protocol designs, such as MintRoute, assumed that intermediate links had stable, independent packet losses, and used this assumption to derive the necessary sampling window for an accurate estimate [46].

But links are highly dynamic. Recent studies by Srinivasan et al. have found that many links are not independent, but bursty [35]. Links are highly dynamic and shift between bursts of very high and very low reception, sometimes on the scale of hundreds of milliseconds. In contrast, protocols settle for beacon rates on the order of 10s of seconds, leading to typical rate mismatches of 2 to 3 orders of magnitude. This means that while low-rate, periodic control packets might observe a reception ratio of 50%, data packets observe periods of 0% and 100%. The periods of 0% cause many retransmissions and failed deliveries.

2.2 Packet Sampling Bias

To sidestep the numerous issues in accurate link estimation with packet counts, some protocols use physical layer information to es-

timate link quality. For example, the standard TinyOS collection protocol, MultiHopLQI, uses a hardware feature of the CC2420 radio to get an approximation of the bit error rate in a received packet. It uses this value to estimate the quality of the link.

The problem with this approach is it only samples on successfully received packets. If a node is loses 50% of the packets on a link due to collisions or link variations, but those that it receives have few bit errors, it will assume the link is good. Figure 1 shows an example trace of this phenomenon. While this trace relates to LQI, the same is true of signal strength (RSSI). Assuming a high signal strength is always a high quality link assues that the signal strength is constant and that there is no interference. Figure 2 shows a 10 second link probe trace from the Mirage testbed, where the RSSI changes by over 10dB and PRR over a 1s window varies from 95% to 0%. A node using periodic RSSI measurements will only observe the good periods and assume the link is excellent.

2.3 Transient Loops

Finally, changes in the topology, such as a link going down, have adverse effects on many routing protocols, causing losses in the data plane or long periods of disconnection while the topology adjusts. In most variations of distributed distance vector algorithms, such changes may result in transient loops which causes packets to be dropped. This is the case even in path-vector protocols like BGP, designed to avoid loop formation [30]. MultiHopLQI, upon detecting a failure, advertises that it has no route, discards its parent, and waits for beacons to give it a new parent. In the meantime, it drops packets. Because beacons are typically on the order of seconds, this can cause a significant outage in the network.

Some protocols are designed to prevent loops from forming altogether. DSDV, for example, uses centralized sequence numbers to synchronize routing topology changes and preclude loops [31]. When a link goes down, the entire subtree whose root used that link is disconnected until a refresh from the root eventually finds an alternate path. Again, even in the absense of loops, there is a relatively long outage in the network.

In both cases, a fundamental problem is that the repairs to the topology happen at the timescale of the control plane maintenance traffic, while the data plane can operate at much faster timescales. Since the data plane has no say in the routing decisions, it has to choose between dropping packets or stop traffic until the topology is fixed.

2.4 Control and Data Plane Separation

Protocols today have a strong separation between the control and data planes, which is rooted in the architecture of their wired counterparts. The control plane is responsible for estimating the quality of links between neighbors, and for forming a topology and selecing routes. Protocols do this by using periodic control packets. The data plane then uses these links and routes to send and forward packets. But the temporal scale of network dynamics is a critical issue, and the control planes and data planes operate at time scales that differ by three or more orders of magnitude: protocols can send data packets every 10ms but send control packets every 10s or more.

One of the key observations of this paper, which we implement and evaluate in Magnet, is that by using valuable information from the data plane to guide important decisions by the control plane, we can address these three issues we presented – link dynamics, packet sampling bias, and transient loops – in a powerful way. By widening the interface between the data plane in to both the link estimation and the route selection components of the data plane in a very controled way, we can make collection robust and efficient in the presence of network dynamics.

3 Design Overview

Magnet is the collection protocol we built to address several of the pitfals we identified in current and past implementations of this routing primitive. In this section we describe the service and abstraction it provides, and give an overview of its main components and how they interact.

3.1 Anycast Collection

The goal of Magnet is to aggregate packets from all nodes in the network to one or more base-stations. To this end, Magnet builds and maintains a topology in which each node has a route to its closest base-station. The protocol is address-free, and all base-stations are treated equally. When a node sends packet through Magnet, it is not specified which basestation will receive the packet, and the protocol attempts to deliver it to the one with the minimum cost. The routing primitive provided is, thus, anycast collection. When there is only one basestation the topology reduces to a single tree that spans the entire network, a pattern commonly referred to as converge-cast.

3.2 Control Plane

The control plane is responsible for estimating the quality of links between neighbors and selecting the routes. Magnet strives to find paths that minimize the Expected Transmissions (ETX) routing metric [8], which is obtained by adding the ETX metric of the links in the path. One of Magnet's central components it the hybrid link estimator that combines information from routing beacons and data forwarding to determine the ETX of links to each neighbor.

Nodes exchange periodic beacons with routing information. Magnet's link estimation component adds unique sequence numbers to these transmitted beacons, such that neighbors can determine when these beacons are lost. The fraction of lost beacons forms a base estimate of the link qualities from neighbors. When it transmits data packets, Magnet uses the information about the number of acknowledged and unacknowledged packets to refine the estimate of link qualities.

Once nodes have link quality estimates, they can build the routing trees. Routing beacons contain a node's path ETX to the closest basestation. Upon receiving routing beacons from neighbors, a node selects a parent that yields the smallest cost path, i.e., the sum of the parent's own ETX and the ETX of the link between the two.

A novel and important feature of Magnet is that the routing beacons are transmitted according to a variable timer, controlled by the Trickle algorithm. Treating the maintenance of the topology as a consistency problem, Trickle will increase the beaconing frequency when an inconsistency is detected, and exponentially decrease the frequency as the topology stabilizes. As we show in the evaluation, this approach ensures agile response to changes while minimizing control traffic overhead. A very important part of the algorithm is how to detect the inconsistencies. Magnet uses the data packets to detect routing inconsistencies, which makes sure detection is timely and does not waste probing resources.

3.3 Data Plane

The data plane is responsible for forwarding packets towards one of the basestations, along the paths defined by the control plane. Magnet forwards packets along the minimum cost routes in a best-effort manner, but tries hard not to drop packets. While an end-to-end protocol is needed to achieve 100% reliability, Magnet routinely achieves 99.9% reliability in our experiments.

To this end the data plane provides invaluable information, at forwarding time, to aid in route repair and in link estimation. Packets being forwarded can be uniquely idenfied by the origin, origin-generated sequence number, and number of hops traversed. With these fields the data plane can detect and remove duplicate transmissions. Packets also carry the smallest cost they have seen to the root, and if this happens to be smaller than the current node's cost, a routing inconsistency is detected. This event immediately triggers the control plane to update its routing topology among the neighbors. Finally, information about the number of retransmissios of a packet directly feeds the hybrid link estimation, again ensuring a more timely and accurate estimate.

3.4 Link Layer Requirements

All together, Magnet's design assumes that the data link layer provides four things:

1. An efficient local broadcast address
2. Synchronous acknowledgements for unicast packets
3. A protocol dispatch field to support multiple layer 3 protocols
4. Single-hop source and destination fields

In the next three sections we describe in detail the link estimation, path selection, and packet forwarding components of Magnet, and how they work together to address the challenges that collection faces in this environment.

4 Link Estimation

Both opportunistic and classical routing protocols depend on neighbor link quality assessments. For Magnet to compute the routes to the sink, the first step is for each node to assess the quality of its links to the neighbors. Accurate link quality estimation requires information from the physical, data link, and network layers from both control and data packets.

Traditionally, wireless routing protocols have used two approaches to estimate link quality. Hardware-assisted link quality estimation uses some function of link quality, signal strength, or similar information provided by the radio as an estimate of link quality. Link layer approaches use periodic beacons that are sent as broadcast packets. Using the sequence numbers in these, nodes in the neighborhood can estimate the inbound link quality. Many implementations of these approaches have several shortcomings:

Estimation bias. When delivery statistics of broadcast control packets are used to estimate the link quality, the statistics does not necessarily accurately predict the delivery performance of unicast data packets which often are of a different size than control traffic and are transmitted using different mechanisms by the MAC.

Packet Sampling bias. Hardware-assisted link estimation technique can use information only on the packets that are received which introduces sampling bias as described in section 2.2.

Asymmetric metric. On some radio platforms and environments, although packets can be delivered in one direction with a few bit errors, transmission in the reverse direction might introduce a large number of bit errors. MultiHopLQI, for instance, only uses the inbound link quality as a proxy for the link quality. Furthermore, large data packets are transmitted in the forward direction while small, and often synchronous, acknowledgments are transmitted in the reverse direction. Beacon-based estimators can not accurately predict the goodness of the link for this asymmetric use of the link.

Not portable. Typically hardware-assisted link estimation techniques are not portable across the platforms. MultiHopLQI uses the radio's LQI indicator. This works reasonably well on platforms with the CC2420 radio but does not work with radios that do not provide LQI value. There have been attempts to port MultiHopLQI to the CC1000 radio, which provides an RSSI measurement instead of LQI. However, it has been found that MultiHopLQI would have to be changed significantly for this port to work because LQI and RSSI, although both are provided by the hardware as an indicator of the signal quality, describe fundamentally different aspects of the signal quality [35].

4.1 Information required for accurate link estimation

To overcome the limitations of the traditional approaches to link estimation, there have been recent proposals [18, 47] to use the information from packet transmissions at the data link layer. Our study indicates that not only do we need to use information from data link layer but also from the physical and network layer to estimate the link quality in low power networks. The physical layer can quantify the state of the medium during incoming packets. The link layer can measure whether packets are delivered and acknowledged. The network layer can provide guidance on which links are the most valuable and should be estimated.

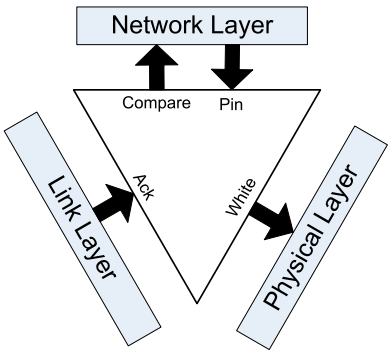


Figure 3. Magnet link estimator, represented by the triangle, uses four bits of information from the three layers. Outgoing arrows represent information the estimator requests on packets it receives. Incoming arrows represent information the layers actively provide.

4.1.1 Physical Layer

The physical layer can provide immediate information on the quality of the decoding of a packet. Such physical layer information provides a fast and inexpensive way to avoid borderline or marginal links. It can increase the agility of an estimator, as well as provide a good first order filter for inclusion in the link estimator table.

As this physical layer information pertains to a single packet and it can only be measured for received packets, channel variations can cause it to be misleading. For example, many links on low power wireless personal area networks are bi-modal [35], alternating between high (100% packet reception ratio, PRR) and low (0% PRR) quality. On such links, the receiver using only physical information will see many packets with high channel quality and might assume the link is good, even if it is missing many packets.

4.1.2 Link Layer

Link estimators such as ETX and MintRoute use periodic broadcast probes to measure incoming packet reception rates. These estimators calculate bidirectional link quality — the probability a packet will be delivered and its acknowledgment received — as the product of the two directions of a link. While simple, this approach is slow to adapt, and assumes that periodic broadcasts and data traffic behave similarly.

By enabling layer 2 acknowledgments and counting every acknowledged or unacknowledged packet, a link estimator can generate much more accurate estimates at a rate commensurate with the data traffic. These estimates are also inherently bidirectional. EAR and ETX use feedback from the link layer for link estimation. Rather than inferring the ETX of a link by multiplying two control packet reception rates, with link layer information on data traffic an estimator can actually *measure* ETX. However, albeit accurate, relevant, and fast, sending data packets requires routing information, which in turn requires link quality estimates. This bootstrapping is best done at lower layers. Also, especially in dense networks, choosing the right set of links to estimate can be as important as the estimates themselves, which can get expensive if done solely at the link layer.

4.1.3 Network Layer

The physical layer can provide a rough measure of whether a link might be of high quality, enabling a link estimator to avoid spending effort or marginal or bad links. Once the estimator has gauged the quality of a link, the network layer can in turn decide which links are valuable for routing and which are not. This is important when space in the link table is limited. For example, geographic routing [17] benefits from neighbors that are evenly spread

in all directions, while the S4 routing protocol [27] benefits from links that minimize distance to beacons. One recent infamous wireless sensornet deployment delivered only 2% of the data collected, in part due to disagreements between network and link layers on what links to use. For this reason, the MintRoute protocol integrates the link estimator into its routing layer. SP provides a rich interface for the network layer to inspect and alter the link estimator's neighbor table. The network layer can perform neighbor discovery and link quality estimation, but without access to information such as retransmissions, acknowledgments, or even packet decoding quality, this estimation becomes slow to adapt and expensive.

4.2 Link Estimator and the Network Stack

Figure 3 shows the interfaces each layer provides to a link estimator. Together, the three layers provide four bits of information: two bits for incoming packets and one bit each for transmitted unicast packets and link table entries.

A physical layer provides a single bit of information. If set, this **white bit** denotes that each symbol in received packet has a very low probability of decoding error. A set white bit implies that during the reception, the medium quality is high. The converse is not necessarily true: if the white bit is clear, then the medium quality may or may not have been high during the packet's reception.

A link layer provides one bit of information per transmitted packet: the **ack bit**. A link layer sets the ack bit on a transmit buffer when it receives a layer 2 acknowledgment for that buffer. If the ack bit is clear, the packet may or may not have arrived successfully.

A network layer provides two bits of information, the **pin bit** and the **compare bit**. The pin bit applies to link table entries. When the network layer sets the pin bit on an entry, the link estimator cannot remove it from the table until the bit is cleared. A link estimator can ask a network layer for a compare bit on a packet. The compare bit indicates whether the route provided by the sender of the packet is better than the route provided by one or more of the entries in the link table. We describe how the link estimator uses the compare bit in Section 4.4 below.

4.3 Interface Considerations

The four bits represent what we believe to be the minimal information necessary for a link estimator. Furthermore, we believe that the interfaces are simple enough that they can be implemented for most systems. For example, radios whose physical layers provide signal strength and noise can compute a signal-to-noise ratio for the white bit, using a threshold derived from the signal-to-noise ratio/bit error rate curve. Physical layers that report recovered bit errors or chip correlation can alternatively use this information. In the worst case, if radio hardware provides no such information, the white bit can never be set.

The interfaces introduce one constraint on the link layer: they require a link layer that has synchronous layer 2 acknowledgments. While this might seem demanding, it is worthwhile to note that most commonly-used link layers, such as 802.11 and 802.15.4, have them. Novel or application-specific link layers must include L2 acknowledgment to function in this model.

The compare bit requires that a network layer be able to tell whether the route from the transmitter of a packet is better than the routes of current entries in the link table. The compare bit does not require that the network layer be able to decide on all packets, merely some subset of them. This implies that some subset of network layer packets, such as routing beacons, contain route quality information.

4.4 A hybrid estimator

We describe a hybrid estimator that combines the information provided by the three layers with periodic beacons in order to provide accurate, responsive, and useful link estimates. The estimator maintains a small table (e.g., 10) of candidate links for which it maintains ETX values. It periodically broadcasts beacons that contain a subset of these links. Network layer protocols can also broadcast packets through the estimator, causing it to act as a layer 2.5 protocol that adds a header and footer between layers 2 and 3.

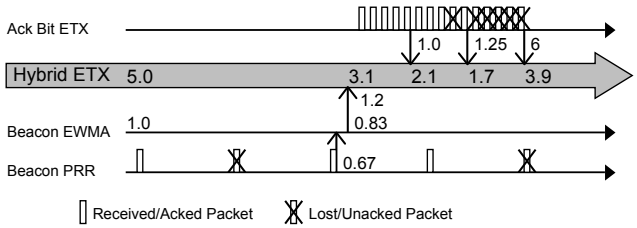


Figure 4. Magnet estimator combines estimates of ETX separately for unicast and broadcast traffic with window sizes of $k_u = 5$ and $k_b = 2$ (for this example) respectively. The latter are first themselves averaged before being combined. We show incoming packets are light boxes, marking dropped packets with an "×". The estimator calculates link estimates for each of the two estimators at the times indicated with vertical arrows.

The estimator follows the basic table management algorithm outlined by Woo et al. [46], with one exception: it does not assume a minimum transmission rate, since it can leverage outgoing data traffic to detect broken links. Link estimate broadcasts contain sequence numbers, which receivers use to calculate the beacon reception rate.

The estimator uses the white and compare bits to supplement the standard table replacement policy. When it receives a network layer routing packet which has the white bit set from a node that is not in the link estimation table, the estimator asks the network layer whether the compare bit is set. If so, the estimator flushes a random unpinned entry from the table and replaces it with the sender of the current packet.

The estimator uses the ack bit to refine link estimates, combining broadcast and unicast ETX estimates into a hybrid value, an approach necessitated by the large variance of traffic volume across different links in the network [18]. We follow the link estimation method proposed by Woo et al. [46], separately calculating the ETX value every k_u or k_b packets for unicast and broadcast packets, respectively. If a out of k_u packets are acknowledged by the receivers, the unicast ETX estimate is $\frac{k_u}{a}$. If $a = 0$, then the estimate is the number of failed deliveries since the last successful delivery. The calculation for the broadcast estimate is analogous, but has an extra step. We use a windowed exponentially weighted moving average (EWMA) over the calculated *reception probabilities*, and invert the consecutive samples of this average into ETX values. These two streams of ETX values coming from the two estimators are combined in a second EWMA, as shown in Figure 4. The result is a hybrid data/beacon windowed-mean EWMA estimator. When there is heavy data traffic, unicast estimates dominate. When the network is quiet, broadcast estimates dominate.

Contrary to most pure broadcast-based estimators, our estimator does not actively exchange and maintain bidirectional estimates using the beacons. Because the ack bit inherently allows the measurement of bidirectional characteristics of links, our estimator can afford to only use the incoming beacon estimates as bootstrapping values for the link qualities, which are refined by the data-based estimates later. This is an important feature, as it decouples the in-degree of the nodes in the topology from the size of the link table.

4.5 Control Frame for Link Estimation

Figure 6(a) shows the format of a Magnet link estimation frame. Magnet sends routing frames on top of a link estimation protocol, which appends data advertising the ETX of beacons from other nodes. For example, if Magnet appends a link estimation record

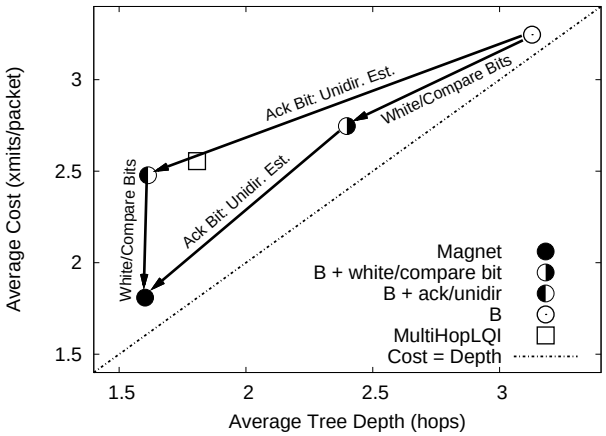


Figure 5. Exploring the link estimation design space: adding the ack bit and/or the white and compare bits to Magnet decreases cost (lower is better) and the average depth of a node in the routing tree.

for node 17 with an ETX of 2.1, this means it estimates that node 17 must transmit 2.1 beacons for it to receive one. Routing beacon ETX values are fixed-point decimal: a value of 32 indicates an ETX of 3.2 while a value of 11 indicates an ETX of 1.1.

The link estimation frame format specifies a two-byte header and a variable-length footer. The header has four bits specifying how many link quality records at in the footer, four reserved bits, and an eight-bit sequence number for estimating incoming link quality. Each link quality record is three bytes, with two bytes for the source link layer address and one byte for the link ETX. If the link ETX is greater than 25.5, the protocol does not consider it a valid link to advertise.

4.6 Evaluation

To understand how the addition of information from the physical, network and link layer improves the performance of purely beacon based estimator, we run four sets of experiment with these link estimators: purely beacon based estimator ("B"), adding white/-compare bit ("B+white/compare"), adding ack bit/unidirectional estimate to the beacon-based estimator ("B+ack/unidir"), and finally Magnet estimator that uses the white, compare, ack bits and unidirectional estimates.

In our comparison of these link estimators, we run collection with with each estimator on the Mirage testbed, using 85 MicaZ nodes with one node set as the basestation. We also ran these experiments on a second testbed, Tutornet, but we do not present detailed results from those experiments because the results are qualitatively similar to the ones from Mirage. Transmit power is set at 0 dBm unless otherwise specified. In each experiment we stagger the boot time of all nodes using a uniform distribution over a range of thirty seconds. Each node sends a collection packet every 8 seconds with some jitter to avoid packet synchronization with other nodes. The workload each node offers is a constant-rate stream of packets sent to a sink. This creates many concurrent flows in the network, converging at the sink. All experiments on Mirage lasted between 40 and 69 minutes. On TutorNet we ran much longer experiments, ranging from 3 to 12 hours.

The primary metric we use to evaluate the performance of these estimators is **cost**: the total number of transmissions in the network for each unique delivered packet. Cost is important as it directly relates to network lifetime. To put cost into perspective, we also look at the **average depth** of the topology trees. If all links are perfect, average depth is a lower bound for cost. The difference between the two is indicative of the quality of the links chosen, as it stems from either retransmissions or dropped packets. Finally, we also look at **delivery rate**, the fraction of unique messages received at the root.

The uppermost-right point of Figure 5 shows the cost and depth of collection using the purely beacon-based estimator "B". Adding unidirectional link estimation and the ack bit to the beacon-based estimator reduces average tree depth by 93%, and reduces cost by 31%. This "B+ack/unidir" estimator decouples in-degree from the link table size, hence the large decrease in depth. In contrast, the Magnet link estimator, that uses all the bits and unidirectional estimate decreases cost further to 55% of the original beacon-based estimator, possibly because of improved parent selection. Adding only the white and compare bits to beacon-based estimator, the resulting "B+white/compare" estimator offers a reductions of 15% in cost and 23% in average depth.

The figure also shows MultiHopLQI's cost and depth in the same testbed for comparison. It is only when we use information from the three layers that Magnet does better than MultiHopLQI. In these experiments, Magnet has 29% lower cost and 11% shorter paths than MultiHopLQI. On another set of experiments on TutorNet, Magnet's cost and average depth were respectively 44% and 9.7% lower than MultiHopLQI's.

5 Path Selection

In this section, we describe how Magnet discovers, advertises, and selects routes. Building on the ETX-based link estimator, a route's cost is the sum of the costs of its links. The Magnet control plane consists of broadcast packets that advertise a node's current parent, its current route cost, and two control bits. Magnet embeds these routing beacons in a link estimation packet, enabling it to seed the link estimation described in Section 4;

5.1 Route Computation

Magnet path selection uses a distance vector routing algorithm, with ETX as the edge weight. Magnet is an anycast protocol: a node is not aware of the address of the sink at the end of its route. This enables Magnet, like many collection protocols, to support multiple sinks with no additional complexity.

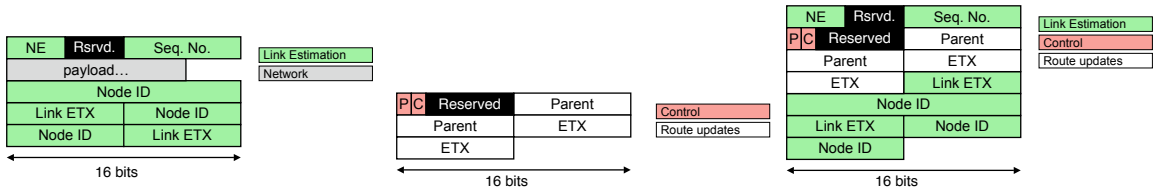
Figure 6 shows the Magnet routing packet format, which contains routing data embedded in a link estimation packet. The routing data advertises the node's current parent and routing cost. It also includes two control bits: the pull bit (P) and the congested bit (C). We discuss the meaning and use of the P bit below and the C bit in Section 6.6. The routing packet also advertises incoming link qualities, which can be used to calculate a bidirectional packet reception rate and avoid asymmetric links.

A sink starts route calculations by advertising a cost of zero. Nodes that receive these beacons with the white bit immediately put the root in their routing table. Otherwise, nodes depend on the root hearing their beacons, putting them in its routing table, and advertising those links in its own beacons. If a node does not have a route, it advertises a cost of infinity, which is defined to be the maximum routing cost value (0xffff) to the sink. Routing cost is a fixed-point number with two significant decimal fractional digits. With 16 bits, this means Magnet can advertise costs in the range [0.00,655.34] and therefore theoretically scale to hundreds of hops.

Note that the Magnet algorithm, like other distance vector routing algorithms, does not guarantee loop-free routing. Transient loops can lead to the loss of a path to the root and consequently packet loss. A protocol that attempts to offer high delivery performance must detect and repair loops quickly. Magnet uses the feedback from data plane to detect loops quickly (Section 6.3) and uses the Trickle algorithm to time its control traffic for rapid loop repair.

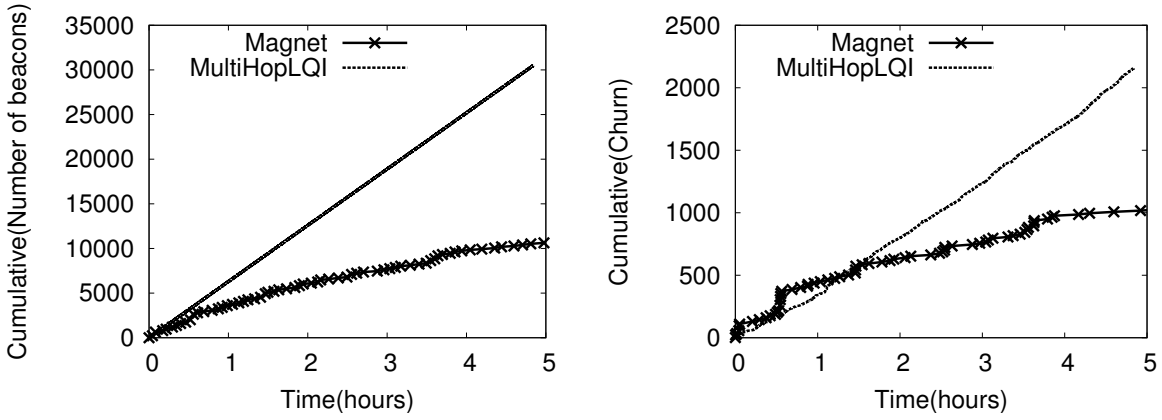
5.2 Route Selection

Magnet seeks to send packets along the minimum cost route to a sink. However, there is a tradeoff to be made between route stability and optimality. In practice, changing routes very quickly and



(a) Link estimation frame (b) Routing frame (c) Complete routing frame

Figure 6. Magnet routing frame format



(a) Control Overhead (b) Number of parent changes (Churn)

Figure 7. Control Overhead for MultiHopLQI and Magnet. Over long run, Magnet transmits significantly fewer routing beacons than MultiHopLQI.

often leads to routing instabilities. Active routes tend to have better estimates due to direct, data-traffic measurements. Sending a data packet to a node whose last advertisement was received a long time ago can be dangerous, as it might currently have a much longer route, such that this creates a routing loop.

Furthermore, link estimation values are fundamentally noisy, as being adaptive requires having a small time averaging window. Therefore, even a perfectly stable link with independent losses will see estimate variations.

Magnet addresses link variability this tradeoff by using hysteresis: it only switches routes if it believes the other route is significantly better than its current one, where "significantly" better is defined as having an ETX at least 1 lower.

5.3 Control Traffic and Timing

In the long term, Magnet relies mostly on data packets to maintain, probe, and improve link estimates and routing state. Beacons, however, form a critical part of routing topology maintenance. First, since beacons are broadcasts, they are the only way that nodes can advertise their presence, and so provide the bootstrapping mechanism for neighbor tables. Second, there are times when nodes must communicate information, such as the fact that their route quality has change significantly, to all of their neighbors.

Because Magnet's link estimator does not require a fixed beaconing rate, Magnet can adjust its beaconing rate based on the *expected* importance of the beacon information to its neighbors. When the routing topology is working well and the routing cost estimates are accurate, Magnet can slow its beaconing rate. However, when the routing topology changes significantly, or Magnet detects a problem with the topology, it needs to quickly inform nearby nodes so they can react accordingly.

Magnet sends routing packets using a variant of the Trickle algorithm [23]. It maintains a beaconing interval which varies between 64ms and one hour. Whenever the timer expires, Magnet doubles it, up to the maximum (1 hour). Whenever Magnet detects an event which indicates the topology needs active maintenance, it resets the timer to the minimum (64ms). If a node has a timer of length τ, then it chooses a random value t within the interval in the range of $[\frac{\tau}{2},\tau)$ and transmits at that time. This randomization prevents collisions among nodes whose timers synchronized on hearing a packet.

Three events cause Magnet to reset its interval to the minimum length:

1. **It is asked to forward a data packet from a node whose ETX is lower than its own.** Magnet interprets this as nodes around it having a significantly out-of-date estimate of its routing cost. It therefore begins beaconing to update its interested neighbors.
2. **Its routing cost decreases significantly.** Magnet advertises this event because it might provide lower-cost routes to nearby nodes. In this case, "significant" is an ETX of 2.
3. **It receives a packet with the P bit set.** The "Pull" bit denotes that a node wishes to hear beacons from its neighbors, e.g., because it has just joined the network and needs to seed its routing table. The pull bit provides a mechanism for nodes to actively request topology information from neighbors.

The first two events are indications that a route cost stored in a routing table may be inconsistent with the actual routing cost of the respective node. In the first case, a node needs to advertise its lower cost because others may wish to use it as a forwarder. In the second case, a node needs to advertise its higher cost because others may wish to stop using it as a forwarder. Since the second condition only matters if actually forwards packets, Magnet relies on data traffic as a trigger. The third and final event is to enable rapid node bootstrapping and network joins.

5.4 Evaluation

We now evaluate the amount by which use of a Trickle timer to drive beacon timing minimizes control overhead, without sacrificing any routing agility. Figure 7(a) shows that Magnet control overhead, although high in the beginning as Magnet probes and discovers the topology, evolves sub-linearly over the long term. This data comes from a seven-hour experiment on the Tutornet testbed. Figure 7(a) also shows the evolution of control overhead of MultiHopLQI, which sends beacons at a fixed interval of 30s. Because this interval is fixed, MultiHopLQI's cost is constant over time. Even though Magnet can send beacons as quickly as every 64ms, allowing it to respond to topology problems within a few packet times, by adapting its control rate it can, over the long term, have a lower control packet cost than a protocol whose adaptation period is 500 times longer.

We plot the number of parent changes in figure 7(b). This figure shows that Magnet has fewer parent changes over time compared to MultiHopLQI. Putting the results from these two figures together, we find that MultiHopLQI changed parent six times for every 100 beacons, while Magnet changed parent 10 times for every 100 beacons. This suggests that Magnet, compared to MultiHopLQI, sends beacons when it is more likely to lead to parent changes in the neighborhood, and thus when it is important to send the beacons.

We now evaluate the agility of Magnet and the evolution of control overhead when there are network dynamics. During one of our experiments, we introduced four new nodes in the network 60 minutes after the start of the experiment. Figure 8(a) shows that the control overhead for nodes in the vicinity of the new nodes increases immediately after the nodes were introduced as beacons are sent rapidly. The beacon rate decays shortly afterward. To understand whether only nodes in the vicinity of the new nodes send the rapid beacons or the entire network, in Figure 8(b), we plot the time at which each node sends a beacon; a point on the graph represents a beacon sent by a node (y-axis) at a given time (x-axis). The figure shows many regions of time and node with frequent beacons because Magnet must continuously deal with link dynamics and routing inconsistencies even without the introduction of new nodes. Even in this live environment, we can see the nodes in the vicinity of the new nodes (node id 30 through 42) sending frequent beacons shortly after 60 minutes. A few minutes after the introduction of the node, the beacon rate falls back to the background level. Thus with Magnet, the scope of rapid beacons and updates due to dynamics is constrained to just the affected part of the network. During this experiment, the new nodes were able to send collection packet to the sink over multi-hop topology within four seconds after their boot-up time.

6 Packet Forwarding

Magnet sends all locally generated and received collection packets to the parent selected using the path selection algorithm. For its packet forwarding to be efficient and reliable, Magnet performs careful buffer management and allocation, transmission timing, congestion detection and rate throttling, and provides feedback to the link estimator using packet transmission statistics and to the path selection algorithm if it detects an inconsistency in the routing metric gradient along the path. In this paper, we focus our discussion on transmission timing, detection of routing inconsistencies, and feedback to the control plane.

6.1 Data Timing

Wireless routing protocols encounter *self-interference*, where a node quickly sending packets can cause collisions at its parent. For a route of nodes n_i, n_{i+1}, n_{i+2}, self-interference can easily occur at n_{i+1} when n_i transmits a new packet at the same time n_{i+1} forwards the previous one [24].

Magnet prevents self interference by rate-limiting its transmissions. In the idealized scenario above where only the immediate children and parent are in the transmission range of a transmitter, if a n_i waits at least 2 packet times after transmitting a packet before transmitting the next one, then it will avoid self-interference, as n_{i+2} will have finished forwarding [45]. In a real network, the packets can be heard a few hops up and down the path so the node typically should wait for more than two packet times to avoid self-interference. However, if it waits for more time than necessary, then idle channel time is wasted.

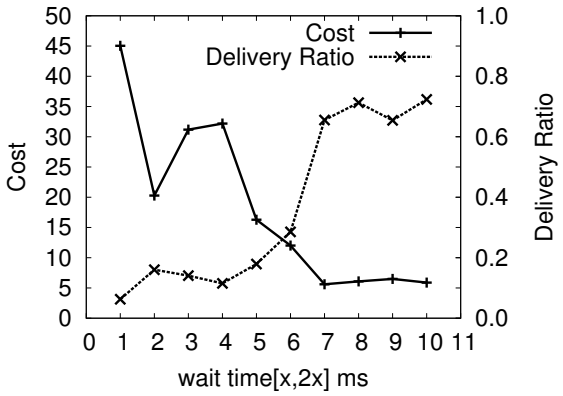
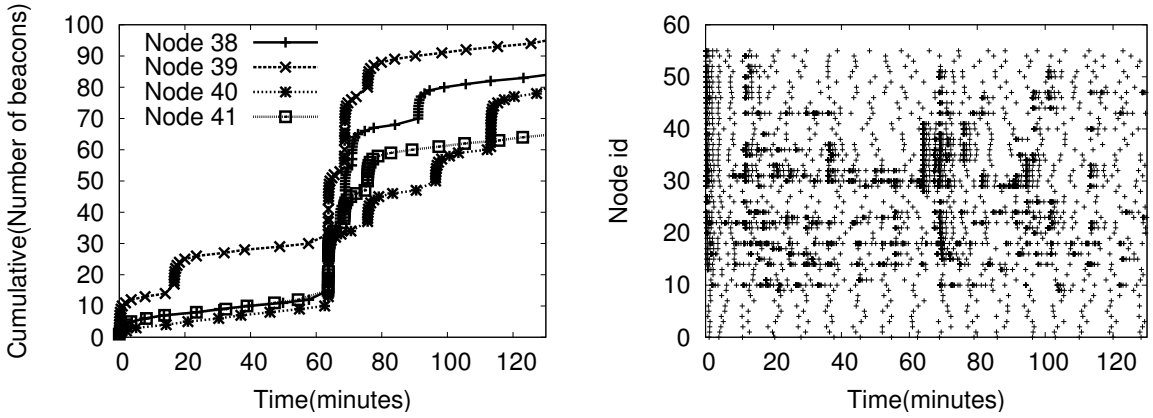


Figure 9. Effect of a per-hop rate-limiting transmit timer on goodput and cost. Wait time is [x,2x] ms.

The transmission wait timer depends on the packet rate of the radio. Here we show how we empirically establish this value for the CC2420 radio. Figure 9 shows the effect of a varying transmission wait timer on a single node flow in the Tutornet testbed. In this experiment, a single node sends packets as fast as it can across multiple hops to the data sink. Even when there is minimal self-interference, the node does achieve a delivery ratio above 77% because there is no end-to-end rate control. Since the node's first hop has a lower ETX than a subsequent hop, it can generate packets faster than the route can support. The transmission timers in Figure 9 range from $[1,2]$ to $[10,20]$ms. At values below $[7,14]$ms, delivery goes down 50% and cost increases 150% due to self-interference. Note that these are not normalized values. However, if the retransmission timer is too large, then the network is idle and goodput can be adversely affected. Based on these results, we conclude that the optimal timer is in the range of 7-14ms for CC2420 radio, which translates to 1-2 pkt times, the wait interval used by Magnet.

6.2 Duplicate Suppression

Collection protocols that use single-hop retransmissions must suppress duplicates created when acknowledgments are lost. Not suppressing duplicates could easily create an exponential number of copies of a packet in a network. Over a small number of hops, this is not a huge issue, but in the presence of transient routing loops, it can quickly overflow all the queues in the loop. Current collection protocols typically suppress duplicates based on link-layer sequence numbers and source addresses. This has the problem that duplicates which traverse two separate routes will not appear as such. Using a combination of an origin and origin sequence number is not sufficient either since a node cannot distinguish a duplicate transmission from a packet that has traversed a transient routing loop. We found that without including THL (see Section 6.6) in the unique packet signature, a network under high routing churn could drop a significant percentage of packets along certain routes (up to 10%). Magnet uses the THL field rather than a link-layer sequence number so that implementations can reorder packets. For example, an implementation designed for low latency might scan across its queue rather than block on the head of the queue, in order to take advantage of packets that arrive but whose acknowledgments are lost.



(a) Number of beacons for selected nodes in the neighborhood of the new node. There is a big jump in control traffic shortly after the new node is introduced and it levels off.

(b) Control traffic sent in the entire network. Many nodes across the network send beacons shortly after the new node is introduced.

Figure 8. Four new nodes are introduced into the network shortly after 60 minutes. The new node transmits a beacon with the PULL bit set. In response, the nodes in the neighborhood reset their Trickle timer. These graphs show how control traffic rate thus adapts to dynamics in the network

Duplicates can often increase the delivery ratio, as their redundancy can prevent packet drops. However, because duplicates are applied uniformly to packets, their cost for their benefit is much higher than more intelligent techniques such as ARQ. If 100% reliability is needed, end-to-end mechanisms may be more appropriate.

A packet signature cache is used to store a 40-bit hash of the packets recently forwarded. Intuition suggests that this cache should be as large as possible. We have found that having a cache size of four slots is enough to suppress most duplicates on the testbeds that we used for our experiments. A larger cache improves duplicate detection but not significantly enough justify larger caches on memory-constrained platforms.

6.3 Routing inconsistencies and loops

A major problem faced by the distance vector routing protocols such as Magnet is the formation of transient routing loops [30]. Loops occur due to inconsistencies in ETX measurements (a child thinks a parent has a lower path ETX than it does) and delay in routing state synchronization with the neighbors. Magnet uses the gradient field in each data frame to deterministically discover routing inconsistencies before the packet has traversed the whole loop. This is in contrast to other protocols, which try to detect repeated packets as a sign of loop and waiting until the next normally-scheduled round of beacons to repair the topology. Any delay in repair may cause a single packet to go through a loop many times. By quickly adjusting in response to routing inconsistencies and pausing data traffic until an update is sent, Magnet can prevent a looping packet from causing a flurry of transmissions

Loops can cause packet drops which increases the cost per delivered packet, an important metric to describe the efficiency of a routing protocol in a low power network. We would invest transmissions on packets that does not get delivered to the sink due to loops. If loops are not repaired quickly, we lose more packets and increase cost of the network while not increasing the delivery.

Magnet uses routing metric non-monotonicity along a path to predict an imminent loop. In contrast, the most popular techniques used in distance vector protocols predict this inconsistency indirectly:

Long paths as a sign of loop. TTL is used to keep track of the number of links a packet has traversed. If a packet traverses enough links to decrement the TTL to 0, that can be an indirect sign that the path has become unusually long due to loops. This method does not detect a loop directly and is sensitive to the TTL number which can be hard to determine for a network of unknown size.

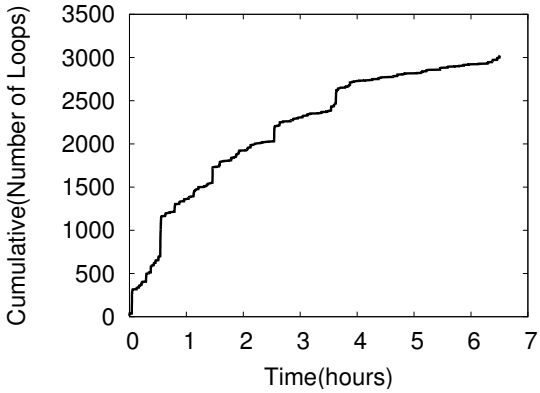


Figure 10. Number of loops over time. Note how there are fewer loops after the network settles down.

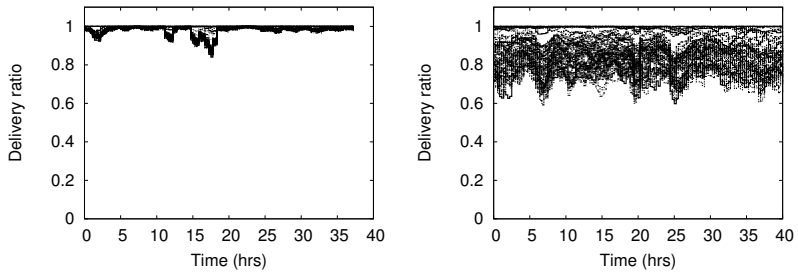
Duplicates as a sign of loop. If there is a routing loop on a path, the packet visits a node multiple times. Thus, if a node receives the same packet twice, that can be due to a loop. The problem is, there can be duplicates due to lost acknowledgments, end-to-end retransmissions, or packets being forwarded on multiple paths that eventually merge. Because there is no way to distinguish between these causes for duplicates from those due to loops, these protocols just drop the packet.

Rather than relying on indirect symptoms of loops, Magnet attempts to directly detect the routing inconsistencies that can cause loops. In a stable network, the routing state across the network is expected to be consistent, which we describe next.

Let p be a path consisting of k links between the node n_0 and the root r. Let n_0 and n_1 be the first and second nodes on this path. n_0 forwards its packet to n_1, n_i to n_{i+1}, and finally n_k to the root r, which is n_{k+1}. For the routing state to be consistent, the following constraint must be satisfied:

$\bigcap_{0..k}^{i} cost(n_i) \geq cost(n_{i+1})$

where *cost* is the delivery cost to the root, estimated using the routing metric.



(a) Delivery Ratio for Magnet (b) Delivery Ratio for MultiHopLQI.

Figure 13. Magnet has consistently higher delivery ratio compared to MultiHopLQI. In figure 13(a) and 13(b), there is one line per node on the testbed; each line shows the delivery ratio of a node over time.

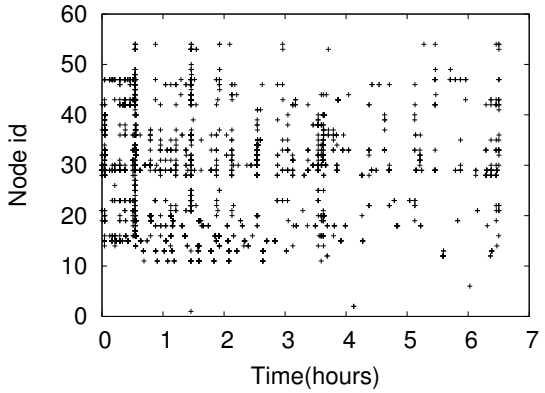


Figure 11. Inconsistent routing states over time across the nodes. Note that these inconsistencies are typically experienced by a set of nodes at any given time. If not fixed quickly, these inconsistencies can lead to loops.

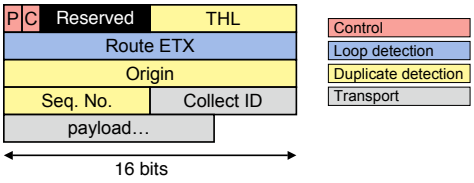


Figure 12. The Magnet data frame format.

Due to network or link dynamics, routing state can become inconsistent. If one of the links suddenly were to degrade in quality, because Magnet employs datapath estimate, the higher loss rate is immediately reflected in its cost along the path to the root. However, this information does not immediately propagate to other nodes in the network. When Magnet forwards a data packet, on each hop, Magnet puts the routing metric for the transmitter. Upon receiving a packet, the receiver compares the routing metric of the transmitter with its own. If the receiver's routing metric is larger, Magnet indicates an inconsistent routing state. Although an inconsistent state is a precursor to a loop, it is possible that this inconsistency has yet to create a loop in the network. Thus Magnet continues to forward the packet to the next hop, albeit at a slower rate so as to not use up all the network capacity while it initiates a more aggressive attempt to synchronize the routing state across the network.

Figure 10 shows the number of inconsistent routes detected by Magnet over time. As time progresses, there are fewer inconsistencies across the network. In the beginning, most of the inconsistencies are due to discovery and initial rounds of path selection. Over time, network dynamics are the dominant cause for routing inconsistency. Figure 11 shows the inconsistencies detected by each node over time. A point on the graph indicates that a node (y-axis) detected an inconsistency at a given time (x-axis). We observe that the inconsistencies are temporally correlated across the nodes and that these inconsistencies are typically constrained to a subset of nodes.

6.4 Dataplane feedback to the routing protocol

Since Magnet identifies routing inconsistency directly and does not need to consider the possibility of long path causing TTL to decrement to 0 or packet duplicates for other reasons making the packet being received again, it can take steps to address the routing inconsistency. It throttles the rate at which packets are forwarded then triggers frequent control traffic to make the routing state consistent across the network by exchanging the most up-to-date link and path ETX values in the neighborhood. For example, if link quality on a path degrades significantly, causing routing inconsistency, Magnet triggers routing beacons to send the new link and path metric values. The nodes in the neighborhood can then update their paths accordingly and select a new parent that has lower cost to the root. Thus the network state becomes consistent.

Magnet uses the Trickle algorithm to efficiently maintain routing consistency across the network. Routing inconsistencies can cause loops and packet drops as the queues eventually fill up so they must be repaired quickly. So, the discovery of a non-monotonic metric along a path resets the Trickle timer causing the node to send beacons rapidly to synchronize the routing states with the neighbors. Discovery of better and more efficient paths are important but not as critical for packet deliveries (unless the current path is not reliable) as fixing routing state inconsistencies. So Magnet uses the beacons sent at large Trickle intervals to slowly discover those potentially more efficient routes.

6.5 Dataplane feedback to the link estimator

The dataplane can provide packet transmission statistics to the link estimator to update the link estimate. These statistics based on data packet transmissions are more accurate than the estimates based on beacon packets. Each time a packet is forwarded, depending on the whether the packet is acknowledged, the dataplane instructs the link estimator to either increase or decrease the perceived quality of the link. More details on the minimalist interface we propose for this interaction is covered in section 4.2.

6.6 Data Frame

Figure 12 shows a Magnet data frame, which is eight bytes long. The data frame shares two fields with the routing frame, the *control* field and the *route ETX* field. The *C* bit operates independently for the two packet types. When the forwarding queue fills up and a node drops a packet, it must set the *C* bit on the next routing frame and the next data frame, allowing the datapath to quickly pass congestion information up the routing tree without requiring a routing frame.

Like the routing frame, the route ETX field of a data frame is fixed point decimal with a range of $0 - 655.35$.

A data frame has an origin address and an 8-bit origin sequence number, which identify unique packets generated by an endpoint. In addition, the dataframe has an 8-bit time has lived, or *THL* field. This field is the opposite of a TTL: it starts at zero at an end point and each hop increments it by one. The three foregoing fields constitute the packet signature. Magnet uses this signature to determine if a packet is a duplicate so that the duplicates can be discarded.

Finally, a Magnet data frame has a one-byte identifier, *CollectID*, for higher-level protocols. All the packets sent by an application or a higher-level protocol has its unique *CollectID* in the data frame. This field acts as an application dispatch identifier, similar to how port number is used by TCP to dispatch the packets to the right application. The *CollectID* allows multiple transport protocols to share a single Magnet stack.

7 Evaluation

Magnet consistently achieves high delivery ratio at a low cost while remaining robust to network dynamics and failures. In this section, we focus our study of those claims.

7.1 Experiment Setup

We have run the TinyOS implementation of Magnet on four different testbeds. Collectively, we have done almost 200 hours of measurements with Magnet, in addition to more than 100 hours of measurements with MultiHopLQI. In this section, unless otherwise stated, we focus on the results from a few experiments on the Tutornet testbed. Tutornet has 56 TelosB nodes in an office building with significant WiFi interference. TelosB motes have CC2420 radio which uses the same frequency space as 802.11 wireless radios. Most other sensor network protocol evaluations on this platform use channel 26 which is the only channel that does not overlap with the 802.11 channels. Although we have done experiments on channel 26, in this paper, we present results from our experiments in channel 16 because this channel overlaps the 802.11 channel 6 which makes it much more challenging to achieve high reliability and robustness at low cost. The interference stresses the protocol and allows us to understand how Magnet would behave in real world deployments where a collection protocol might have to contend for channel with many other unknown and unavoidable devices and protocols such as WiFi, ZigBee, and Bluetooth that use the same frequency space. In the experiments in this section, the collection application that runs on each node generates a packet every 8s that the collection protocol, Magnet or MultiHopLQI, attempts to deliver to a single sink.

7.2 Results

We briefly describe the results from a few selected experiments. Most of these experiments last several hours so the statistics are computed over hundreds of thousands of packets.

7.2.1 Delivery Ratio

Magnet typically achieves two 9's of reliability across a wide range of environments and platforms. Figure 15 shows that Magnet delivers consistently high delivery ratio for packets from all the nodes in the network across multiple runs of varying lengths. In contrast, MultiHopLQI, generally achieves lower delivery ratio, shows variable delivery ratios across the nodes and the experiments. We found out that the difference in delivery ratio for nodes close and far from the root is greater in MultiHopLQI than in Magnet. Although we do not present results from other experiments, we have found that typical MultiHopLQI delivery ratio can range from 70% to 100% while Magnet delivery ratios are in a much narrower range of 97% to 100%.

To understand the consistency of delivery ratio over time, in Figure 13(a), we show the result from one experiment when we ran Magnet for over 37 hours with an application that injected one collection packet every 8 seconds. The delivery ratio remains consistently high over the duration of the experiment. Figure 13(b) shows the result from that experiment. Although the average delivery ratio was 85% in this experiment, the delivery ratio shows high variability over time, occasionally dipping to 58% for some nodes.

7.2.2 Network Efficiency

A protocol that requires a large number of transmissions is not well-suited for duty-cycled network. We measure data delivery efficiency using the cost metric which accounts for all the control and data transmissions in the network normalized by the packets received at the sink. This metric gives a rough measure of the energy spent delivering a single packet to the sink. Figure 14(c) compares the delivery cost for Magnet and MultiHopLQI. Magnet cost is 24% lower than that of MultiHopLQI. The figure also shows that the control overhead for Magnet occupies a tiny fraction (2.2 %) of the total overhead as opposed to almost 8.4% for MultiHopLQI. Although, paths selected by Magnet is often longer than those selected by MultiHopLQI, Magnet delivers packets with lower cost due to lower control overhead and selection of paths that incur fewer retransmissions, while achieving consistently higher delivery ratio than MultiHopLQI.

7.2.3 Robustness

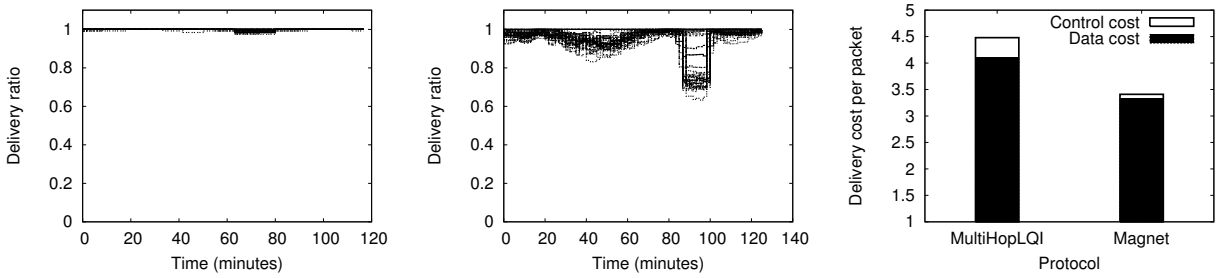
Magnet can quickly adapt to network dynamics and failures. To understand how quickly it can adapt to node failures and the resulting transient loss of performance, we ran Magnet for two hours with an application sending packets every 8s. But shortly after 60 minutes, we removed ten most active forwarding nodes from the network. Removal of these hub nodes could result in temporary packet delivery disruption for a large number of nodes. Figure 14(a) plots the delivery ratio of each node on the testbed over time. The figure shows no significant change in delivery ratio due to the disruption. Figure 14(b) shows the result of a similar experiment with MultiHopLQI. In this experiment, we removed the ten most active forwarding nodes 80 minutes after the start of the experiment. The resulting disruption, which caused the delivery ratio of some nodes to reach as low as 60%, lasted about ten minutes. Magnet quickly detects and recovers from the failures: when data transmission fails for a few times, it updates the link quality, which triggers new parent selection. The recovery completes within a timescale of forwarding a few packets, typically few tens of milliseconds.

7.2.4 Energy Profile

Finally, we wanted to make sure we can run Magnet in an extremely low power networks. For this, we used an experimental mote platform with a TI MSP430 microcontroller and a CC1100 [3] radio. Unlike CC1000 and CC2420 radios, which are used on most mote platforms, CC1100 has a Wake-On-Radio capability which allows it sense the channel without the rest of the node being powered. If it detects a packet, it can wake up the node for processing. This Wake-On-Radio capability makes CC1100 suitable for deployments in ultra low duty-cycled networks.

We setup a 20-node network in a 100x100ft space in a private research building. Borrowing the ideas from BMAC [32], we wrote a MAC for CC1100. Our MAC differs from BMAC in that it uses the built-in Wake-On-Radio capability to detect the presence of packets instead of CPU-driven CCAs. The parameter *sleep interval* determines the preamble length for transmissions, exactly like in BMAC. We instrumented a node close to the root by connecting a precision 1 Ω resistor in series with the battery to measure the voltage across the resistor. We sampled the voltage at 50 KHz with a high precision 24-bit ADC using a data acquisition unit. We later converted this voltage to energy and extrapolated to per day energy consumption.

We perform a total of 15 experiments for energy profiling changing the MAC sleep interval and application data generation interval. For each experiment, we let the network warm up for about 15 minutes. We then use a dissemination protocol to request the nodes to start sending data at a given message interval. We collect the data

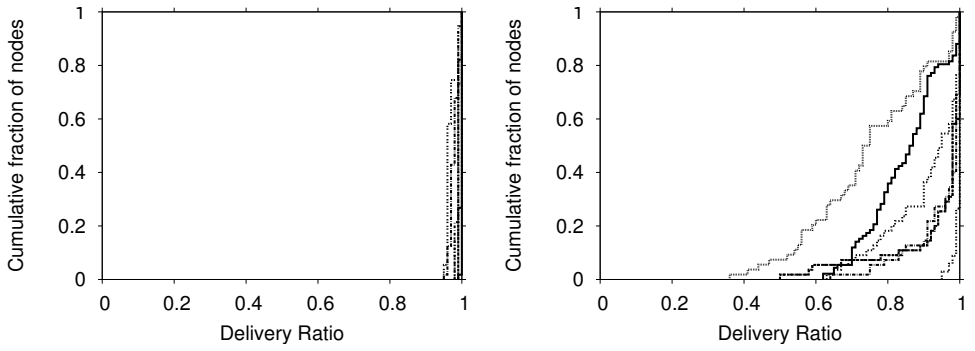


(a) When ten most active forwarding nodes were removed from the network after 60 minutes, with Magnet, we did not observe any significant disruption in packet delivery.

(b) When ten most active forwarding nodes were removed after 80 minutes, with Mul- tiHopLQI, the disruption lasted in the order of ten minutes.

(c) Magnet delivers packets at 24% lower cost than with MultiHopLQI.

Figure 14. Robustness and Efficiency of Magnet and MultiHoplQI.



(a) Delivery ratios for Magnet

(b) Delivery ratios for MultiHopLQI

Figure 15. Delivery ratios achieved by Magnet and MultiHopLQI over 7 independent runs of each, ranging from 1 to 18hrs.

for about 15 minutes. In each experiment, we observed that Magnet was forming a multi-hop topology but only delivering about 95% of the collection packets at the sink. Though this is lower than the two 9's of delivery ratio we typically achieve in an always-on network, upon careful analysis of traces from these experiments, we found that most of the packet loss could be attributed to the limitation of this experimental MAC - problems with acknowledgment timing and channel access during contention.

The platform consumes 341.6 mAh/day in idle mode without duty-cycling. The result from figure 16 shows that we can run the full Magnet stack for as little as 7.2 mAh/day, compared to 59.2 mAh/day for MultiHopLQI. This result suggests that Magnet with a properly-designed low power hardware platform can be used in long lasting deployments: even with a moderately-rapid (for a low power network) message interval of 240s, two AA batteries with a total of 5000 mAH capacity can be used to operate a node for more than 400 days. This result is significant because our evaluation takes into account the cost for running a full network stack consisting of dissemination and Magnet protocols. Magnet's low energy profile is possible only because it selects efficient paths, takes measure to avoid unnecessary control traffic, and actively monitors the link and path quality using the feedback from the data plane. Reduction in control traffic is especially important in these networks because broadcast packets must be transmitted with long preambles.

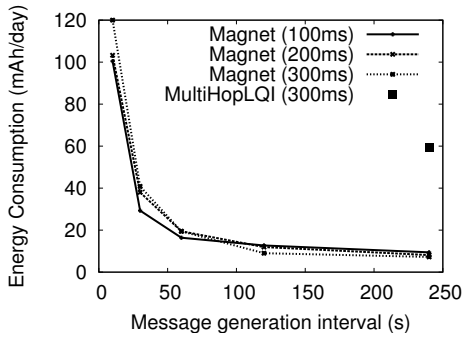


Figure 16. Energy consumption for Magnet and MultiHopLQI for 100ms-300ms sleep intervals.

8 Related Work

The choices we made in the design of Magnet draw on experience with the few other collection layer implementations such as MultihopLQI [37], MintRoute [46], and Boomerang [1]. We also draw on solutions to the well-studied problems of link estimation, routing, and congestion control, as described below.

Magnet also borrows from work on reliable point-to-point transport protocols for wireless sensor networks [20, 34, 36, 40] which seek to maximize data delivery throughput by timing transmissions on a path such that a pipelining effect occurs. This technique motives our use of the forwarding timer in Section 6.1.

At a high level, Magnet combines the elements of proactive and reactive routing paradigms, as it proactively maintains (at low cost) a rough approximation of the best routing gradient, and makes an effort to improve paths which data traffic traverses. Magnet beacons are driven using the Trickle algorithm [23], which allows Magnet to quickly discover new nodes or recover from failures, but uses long beacon intervals when the network is stable. Thus Magnet approxi-

mates beaconless routing [47] in stable and static networks without sacrificing agility and the capability for new node discovery.

Recent work shows that a good way of constructing routes in a wireless mesh network is to minimize either the expected number of transmissions (ETX [8]) or a bandwidth-aware function of the expected number of transmissions (ENT [9]) along a path [46], instead of simply the number of hops a packet must travel along any path. We draw on this work, implementing Magnet with ETX as the routing metric.

Predicting how well a particular link will deliver packets is a necessary step in many collection protocols. In particular, assigning each link a packet reception rate (PRR) is difficult due to a large "grey area" in sensor networks [48] and a lack of correlation between PHY-layer information and PRR in this grey area [35, 48]. A further factor encouraging PHY-independence is our desire to build a cross-platform collection layer.

A number of routing layers [37,38] and studies [35] propose link quality estimates from the PHY as PRR estimators, and we describe the pitfalls of these approaches and how well they perform against Magnet in Section 4.

The link estimators of many routing layers use PRR estimators based directly on packet loss statistics. MintRoute [46] uses a windowed mean with EWMA (WMEWMA) estimator to estimate link quality. In separate work, Woo and Culler also consider numerous other estimators [44] such as EWMA, flip-flop EWMA (discussed below), and windowed moving average, and find them inferior to WMEWMA. Like MintRoute, we use WMEWMA estimators, but unlike MintRoute and other known prior work, we separate link estimates based on data and beacon traffic, yielding a mechanism similar to the flip-flop EWMA estimator discussed below.

The EAR link quality measurement framework [18] uses a combination of passive, active, and cooperative techniques to measure link qualities in a mesh network. The Magnet link estimator presented in Section 4 adapts some of EAR's techniques to Magnet, namely passive and active link monitoring. Magnet's purpose, however, is to estimate link qualities to construct a collection tree. This is a distinct problem from routing in a mesh network, because the former typically has fewer bandwidth requirements while the later requires all-paths routing.

"Learn on the fly" [47] uses MAC latency until the successful transmission of a packet as link metric, and although can be used with ETX path metric, focuses on a discussion of a beaconless geographic routing with static nodes. This work, like other proposals [13,15,22], and ours, share the the same philosophy, using information from the link layer for accurate link estimation. The Magnet estimator, in addition to the information from the link layer, also uses information from the physical and network layers using well-defined and narrow interfaces to the physical, network, and data link layers. Our work borrows ideas from these proposals for link estimation but also proposes mechanisms to make collection efficient using networking techniques such as inconsistency detection and control traffic timing.

Also in the domain of mesh wireless networks, Noble et al. [19] propose a Flip-Flop Filter for link estimation. The flip-flop filter switches between an agile EWMA and a stable EWMA depending on whether new data falls within three times the standard deviation of the sample data (the 3σ rule). Woo and Culler [44] evaluate the EWMA flip-flop filter, and find that it does not provide an advantage for the problem of link estimation in low-power wireless networks.

A large body of work in the wireless sensor networking community examines how to mitigate the congestion that occurs when collection traffic concentrates in the vicinity of a sink [10, 16, 33, 41, 42]. Magnet facilitates the use of such higher-layer protocols, but does not replace their functionality. The use of transmit timers (described above in Section 6.1) is the only exception to this rule.

RAP [25] is a network architecture for real-time collection in wireless sensor networks. Unlike other collection protocols, RAP attempts to deliver data to a sink within a time constraint, or not at all: a different task as compared to collection. However, RAP uses similar mechanisms as Magnet, such as MAC priorities and queueing mechanisms.

Ee *et al.* [11] describe a decomposition for the network layer in sensor networks into forwarding and routing modules, with a narrow interface between them. While our work agrees to a large degree with that decomposition, we find that augmenting the interface between the two modules to allow the data plane to trigger updates in the routing module provides a large benefit. We also augment the interface between the two modules and the link estimator.

9 Conclusion

Magnet is a collection protocol that offers 97-99.9% packet delivery in highly dynamic environments while sending up to 25% fewer packets than existing approaches. It is highly robust to topology changes and failures. Furthermore, it places a minimal four expecations on the link layer, allowing it to run on a wide range of platforms without any need for fine-tuning parameters: we have tested the implementation described in this paper on Telos, mica2, micaz, and Blaze. Interoperable versions also exist in Java (for the SunSPOT [2]) and on MantisOS [7].

Magnet achieves these results and flexibility by relaxing the traditionally hard separation between the control and data planes. This integration allows Magnet to use data traffic to validate the correctness of the existing topology as well as quickly detect problems. Control traffic is necessary to repair the topology, however, so using a Trickle timer to regulate control traffic allows Magnet to recover very quickly yet have a low steady-state cost. Integrating information across the physical, link, and network layers allows Magnet to avoid the edge cases that each layer is vulnerable to by itself.

Collection trees are a basic building block for sensor network protocols and systems, yet historically their performance has often been problematic. Our hope is that having a robust, efficient, and portable collection layer will enable research on higher layers by saving developers and implementers the continual effort of revisiting issues such as link estimation, topology discovery, and route selection.

10 References

[1] Moteiv Corp.: Boomerang. `http://www.moteiv.com/software`.

[2] Sun Microsystems. Project Sun SPOT. `http://www.sunspotworld.com/`.

[3] Texas Instruments, CC1100 Data Sheet. `http://focus.ti.com/lit/ds/symlink/cc1100.pdf`, 2003.

[4]

[5]

[6]

[7] BHATTI, S., CARLSON, J., DAI, H., DENG, J., ROSE, J., SHUCKER, A. S. B., GRUENWALD, C., TORGERSON, A., AND HAN, R. MANTIS OS: An Embedded Multithreaded Operating System for Wireless Micro Sensor Platforms. *ACM/Kluwer Mobile Networks and Applications (MONET) Special Issue on Wireless Sensor Networks 10*, 4, 563–579.

[8] COUTO, D. S. J. D., AGUAYO, D., BICKET, J., AND MORRIS, R. A High-Throughput Path Metric for Multi-Hop Wireless Routing. In *Proc. of the ACM MobiCom Conf.* (San Diego, CA, Sept. 2003).

[9] DRAVES, R., PADHYE, J., AND ZILL, B. Comparison of routing metrics for static multi-hop wireless networks. In *Proc. of the ACM SIGCOMM Conf.* (Portland, OR, Aug. 2004), pp. 133–144.

[10] EE, C. T., AND BAJCSY, R. Congestion control and fairness for many-to-one routing in sensor networks. In *Proc. of the ACM SenSys Conf.* [4], pp. 148–161.

[11] EE, C. T., FONSECA, R., KIM, S., MOON, D., TAVAKOLI, A., CULLER, D., SHENKER, S., AND STOICA, I. A modular network

layer for sensornets. In *USENIX'06: Proceedings of the 7th conference on USENIX Symposium on Operating Systems Design and Implementation* (2006), USENIX Association, pp. 18–18.

[12] EE, C. T., RATNASAMY, S., AND SHENKER, S. Practical Data-Centric Storage. In *Proc. of the USENIX NSDI Conf.* (San Jose, CA, May 2006).

[13] FONSECA, R., GNAWALI, O., JAMIESON, K., AND LEVIS, P. Four Bit Wireless Link Estimation. In *Hotnets-VI* (Atlanta, GA, Nov. 2007).

[14] FONSECA, R., RATNASAMY, S., ZHAO, J., EE, C. T., CULLER, D., SHENKER, S., AND STOICA, I. Beacon Vector Routing: Scalable Point-to-Point Routing in Wireless Sensornets. In *Proc. of the USENIX NSDI Conf.* [5].

[15] HE, T., STANKOVIC, J., LU, C., AND ABDELZAHER, T. Speed: a stateless protocol for real-time communication in sensor networks. *Proc. of the 23rd Interation Conference on Distributed Computing Systems* (19-22 May 2003), 46–55.

[16] HULL, B., JAMIESON, K., AND BALAKRISHNAN, H. Mitigating congestion in wireless sensor networks. In *Proc. of the ACM SenSys Conf.* [4], pp. 134–147.

[17] KARP, B., AND KUNG, H. T. GPSR: greedy perimeter stateless routing for wireless networks. In *International Conference on Mobile Computing and Networking (MobiCom 2000)* (Boston, MA, USA, 2000), pp. 243–254.

[18] KIM, K.-H., AND SHIN, K. On Accurate Measurement of Link Quality in Multi-hop Wireless Mesh Networks. In *Proc. of the ACM MobiCom Conf.* (Los Angeles, CA, Sept. 2006), pp. 38–49.

[19] KIM, M., AND NOBLE, B. Mobile Network Estimation. In *Proc. of the ACM MobiCom Conf.* (Rome, Italy, July 2001), pp. 298–309.

[20] KIM, S., FONSECA, R., DUTTA, P., TAVAKOLI, A., CULLER, D., LEVIS, P., SHENKER, S., AND STOICA, I. Flush: a reliable bulk transport protocol for multihop wireless networks. In *Proc. of the ACM SenSys Conf.* (2007), ACM, pp. 351–365.

[21] LANGENDOEN, K., BAGGIO, A., AND VISSER, O. Murphy loves potatoes: Experiences from a pilot sensor network deployment in precision agriculture. In *14th Int. Workshop on Parallel and Distributed Real-Time Systems (WPDRTS)* (apr 2006), pp. 1–8.

[22] LEE, S., BHATTACHARJEE, B., AND BANERJEE, S. Efficient geographic routing in multihop wireless networks. In *MobiHoc '05: Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing* (New York, NY, USA, 2005), ACM, pp. 230–241.

[23] LEVIS, P., PATEL, N., CULLER, D., AND SHENKER, S. Trickle: A self-regulating algorithm for code maintenance and propagation in wireless sensor networks. In *Proc. of the USENIX NSDI Conf.* (San Francisco, CA, Mar. 2004).

[24] LI, J., BLAKE, C., COUTO, D. S. D., LEE, H. I., AND MORRIS, R. Capacity of Ad Hoc wireless networks. In *Proc. of MobiCom* (2001), ACM, pp. 61–69.

[25] LU, C., BLUM, B. M., ABDELZAHER, T. F., STANKOVIC, J. A., AND HE, T. RAP: A Real-Time Communication Architecture for Large-Scale Wireless Sensor Networks. In *Proc. of the IEEE RTAS Symposium* (San Jose, CA, September 2002).

[26] MAINWARING, A., POLASTRE, J., SZEWCZYK, R., CULLER, D., AND ANDERSON, J. Wireless Sensor Networks for Habitat Monitoring. In *Proceedings of the ACM International Workshop on Wireless Sensor Networks and Applications* (Sept. 2002).

[27] MAO, Y., WANG, F., QIU, L., LAM, S., AND SMITH, J. S4: Small State and Small Stretch Routing Protocol for Large Wireless Sensor Networks. In *Proc. of the USENIX NSDI Conf.* (Cambridge, MA, Apr. 2007).

[28] MUSALOIU-E., R., LIANG, C.-J., AND TERZIS, A. Koala: Ultra-low power data retrieval in wireless sensor networks. In *Proc. of the International Conference on Information Processing in Sensor Networks (IPSN 2008)* (2008).

[29] PAEK, J., AND GOVINDAN, R. Rcrt: rate-controlled reliable transport for wireless sensor networks. In *Proc. of the ACM SenSys Conf.* (New York, NY, USA, 2007), ACM, pp. 305–319.

[30] PEI, D., ZHAO, X., MASSEY, D., AND ZHANG, L. A study of bgp path vector route looping behavior. In *ICDCS '04: Proceedings of the 24th International Conference on Distributed Computing Systems (ICDCS'04)* (Washington, DC, USA, 2004), IEEE Computer Society, pp. 720–729.

[31] PERKINS, C., AND BHAGWAT, P. Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers. *Computer Communication Review* (October 1994).

[32] POLASTRE, J., HILL, J., AND CULLER, D. Versatile low power media access for wireless sensor networks. In *Proc. of the ACM SenSys Conf.* [4], pp. 95–107.

[33] RANGWALA, S., GUMMADI, R., GOVINDAN, R., AND PSOUNIS, K. Interference-aware fair rate control in wireless sensor networks. In *Proc. of the ACM SIGCOMM Conf.* (Pisa, Italy, Aug. 2006), pp. 63–74.

[34] SANKARASUBRAMANIAM, Y., ÖZGÜR AKAN, AND AKYILDIZ, I. ESRT: Event-to-Sink Reliable Transport in Wireless Sensor Networks. In *Proc. of the ACM Mobihoc Conf.* (Annapolis, MD, June 2003), pp. 177–189.

[35] SRINIVASAN, K., DUTTA, P., TAVAKOLI, A., AND LEVIS, P. Some implications of low power wireless to ip networking. In *The Fifth Workshop on Hot Topics in Networks (HotNets-V)* (Nov. 2006).

[36] STANN, F., AND HEIDEMANN, J. RMST: Reliable Data Transport in Sensor Networks. In *Proc. of the IEEE SNPA Workshop* (Anchorage, AK, May 2003), pp. 102–112.

[37] TINYOS. MultiHopLQI. `http://www.tinyos.net/tinyos-1.x/tos/lib/MultiHopLQI`, 2004.

[38] TOLLE, G., AND CULLER, D. Design of an application-cooperative management system for wireless sensor networks. In *Proceedings of Second European Workshop on Wireless Sensor Networks(EWSN 2005)* (2005).

[39] TOLLE, G., POLASTRE, J., SZEWCZYK, R., CULLER, D. E., TURNER, N., TU, K., BURGESS, S., DAWSON, T., BUONADONNA, P., GAY, D., AND HONG, W. A macroscope in the redwoods. In *Proc. of the ACM SenSys Conf.* [6], pp. 51–63.

[40] WAN, C.-Y., CAMPBELL, A., AND KRISHNAMURTHY, L. PSFQ: a Reliable Transport Protocol for Wireless Sensor Networks. In *Proc. of the ACM WSNA Workshop* (Atlanta, GA, 2002), pp. 1–11.

[41] WAN, C.-Y., EISENMAN, S., AND CAMPBELL, A. CODA: Congestion Detection and Avoidance in Sensor Networks. In *Proc. ACM SenSys* (Nov. 2003), pp. 266–279.

[42] WAN, C. Y., EISENMAN, S., CAMPBELL, A., AND CROWCROFT, J. Siphon: Overload Traffic Management using Multi-Radio Virtual Sinks. In *Proc. of the ACM SenSys Conf.* [6], pp. 116–129.

[43] WERNER-ALLEN, G., SWIESKOWSKI, P., AND WELSH, M. Fidelity and Yield in a Volcano Monitoring Sensor Network. In *USENIX Symposium on Operating Systems Design and Implementation* (Seattle, WA, Nov. 2006).

[44] WOO, A., AND CULLER, D. Evaluation of Efficient Link Reliability Estimators for Low-Power Wireless Networks. Tech. Rep. CSD-03-1270, UC Berkeley, May 2004.

[45] WOO, A., AND CULLER, D. E. A transmission control scheme for media access in sensor networks. In *Proceedings of the seventh annual international conference on Mobile computing and networking* (Rome, Italy, July 2001).

[46] WOO, A., TONG, T., AND CULLER, D. Taming the underlying challenges of reliable multihop routing in sensor networks. In *Proc. ACM SenSys* (Los Angeles, CA, Nov. 2003), pp. 14–27.

[47] ZHANG, H., ARORA, A., AND SINHA, P. Learn on the fly: Data-driven link estimation and routing in sensor network backbones. In *Proc. of the ACM MobiCom Conf.* (Los Angeles, CA, Sept. 2006).

[48] ZHAO, J., AND GOVINDAN, R. Understanding packet delivery performance in dense wireless sensor networks. In *Proc. ACM SenSys* (Los Angeles, CA, Nov. 2003), pp. 1–13.