Four-Bit Wireless Link Estimation

SING Technical Report SING-07-00

Rodrigo Fonseca, Omprakash Gnawali, Kyle Jamieson, Philip Levis
UC Berkeley, Univ. of Southern California, MIT CSAIL, Stanford Univ.
rfonseca@cs.berkeley.edu, gnawali@usc.edu, jamieson@csail.mit.edu, pal@cs.stanford.edu

Abstract

We consider the problem of estimating link quality in an ad-hoc wireless mesh. We argue that estimating links well requires combining information from the network, link, and physical layers. We propose narrow, protocol-independent link estimation interfaces for the layers, which in total provide four bits of information: 1 from the physical layer, 1 from the link layer, and 2 from the network layer. We present a link estimator design with these interfaces that reduces packet delivery costs by up to 44% over current approaches and maintains a 99% delivery ratio over large, multihop testbeds.

1 Introduction

Accurate link quality estimates are a prerequisite for efficient routing in wireless mesh networks: poor link estimates can cause a 200% or greater slowdown in network throughput [8]. Despite its importance, link estimation remains an open problem, in part because many factors conspire to make it challenging, such as the prevalence of intermediate-quality links [21], the time-varying nature of a wireless channel [17], multipath inter-symbol interference [6], link asymmetries [13], and hardware variations [22]. Furthermore, the physical, link, and network layers each have valuable information that can improve estimates, such as channel quality, packet delivery ratios, route utility, and acknowledgments. The complexity of this design space, combined with the rich information that certain chipsets or protocols can provide, has led many protocols to use *cross-layer* design, where each layer freely shares protocol-specific information in order to improve performance.

In this paper, we propose a different approach. We distill the feedback provided by the physical, link, and network layers for accurate link estimation to narrow interfaces. All together, these interfaces provide 4 bits of information: 1 from the physical layer, 1 from the link layer, and 2 from the network layer. These bits of information are protocol independent, thereby keeping layers decoupled and avoiding unforeseen dependencies that hinder network evolution.

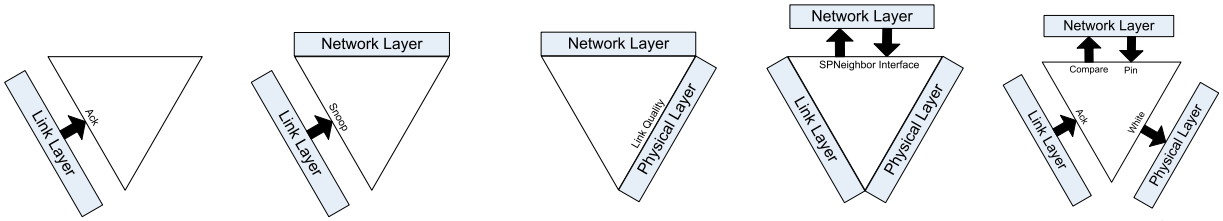
To examine the efficacy of this 4-bit approach, we consider it in a notoriously-difficult class of wireless mesh, wireless sensornets. Unlike higher-power wireless meshes, RAM limitations mean wireless sensornets cannot store state for all possible neighbors. This limitation requires that routing IP in these meshes (e.g., with 6lowpan [1]) requires good route summarization [18]. Therefore, link estimation accuracy is not the only concern: an estimator must also choose good neighbors to estimate.

Each layer in the protocol stack can contribute towards these goals. From the physical layer we can measure channel quality during a packet. Not all packets are equal: a packet with few bit errors is more likely to be from a good link than one which has many bit errors. These physical layer measurements are fast and cheap, enabling a link estimator to avoid spending effort on marginal or poor links [5]. We can distill this down to the white bit, which denotes whether the channel quality during a received packet was high.

From the link layer, we can measure whether packets are delivered and acknowledged. One problem faced by broadcast probe-based estimators, such as ETX [8] and MintRoute [20], is that they decouple link estimation from data traffic: if a link goes bad and packets are lost, the link estimate will not reflect this change until the next routing beacon is dropped. We can distill this down to the ack bit, which denotes whether the node received a layer 2 acknowledgment in response to a transmission.

From the network layer, we can learn which links are the most valuable for higher-layer performance. Without layer 3 information, estimators may select links which form circuitous routes, or in the worst case, which disconnect a network. At least one wireless sensornet deployment has failed due to inconsistency between layer 2 and layer 3 link tables [12]. We can distill these concerns down to two bits: the pin bit, which tells the estimator to not evict a link because it is in use, and the compare bit, which the estimator can use to ask the network layer if a link looks promising.

This paper makes three research contributions. First, in Section 2, we identify valuable information each layer can provide for link estimation and experimentally diagnose failure cases each layer cannot identify. Second, in Section 3, we define a set of narrow interfaces that provide information from each layer to a link estimator. Third, we describe a prototype estimator that uses these interfaces and evaluate its improvement over current approaches in Section 4. Even though the interfaces provide only a total of four bits of information, testbed experiments show that our estimator outperforms existing cross-layer approaches. We compare our estimator with MultihopLQI, the current state of the art estimator used by many sensornet protocols and systems [2, 9, 14, 16, 19]. On the Intel mirage testbed [7], our link estimator reduces packet delivery costs by 29% while maintaining a 99.9% delivery ratio in comparison to Multi-



(a) EAR, ETX (b) MintRoute (c) MultihopLQI (d) SP (e) Four Bit

Figure 1. A link estimator, represented by the triangle in the center of each figure, interacts with up to three layers. Attached boxes represent unified implementation. Outgoing arrows represent information the estimator requests on packets it receives. Incoming arrows represent information the layers actively provide.

hopLQI's 93%. On the USC Tutornet testbed, our link estimator reduces packet delivery costs by 44% while maintaining a 99% delivery ratio in comparison to MultihopLQI's 85%. Furthermore, as this estimator is independent of the three layers, it can be easily incorporated into a wide variety of protocols.

These results provide strong evidence that link estimators can be decoupled from particular layer implementations yet remain efficient and accurate. Decomposing link estimation in this way simplifies network stacks and promotes protocol evolution and interoperability.

2 Layer Limitations

A link estimator should be accurate and efficient. It should provide good estimates of link qualities, and be agile in detecting changes, all the while minimizing memory requirements and overhead traffic. Each of the physical, link, and network layers can provide valuable information for the link estimator, as demonstrated by previous work (c.f. Figure 1). We argue that a link estimator should use information from *all* three layers to best achieve these goals, not only because each layer can provide information that is unique or much more more inexpensively obtained, but also because there are different link conditions that some layers can detect while others cannot. The physical layer's per-packet channel quality assessment cannot always detect channel temporal variations. While the link layer can accurately measure ETX, it cannot inexpensively decide which links to estimate. The network layer knows which links are most useful for routing, but estimating link qualities at the network layer is inefficient and slow to adapt.

To ground our discussion, we will look into a class of multihop traffic called *collection*, in which multiple nodes send data in anycast fashion to one of possibly many basestations. This is the most prevalent traffic pattern in wireless sensor networks. Our results, though, are also applicable to more general any-to-any multihop traffic.

As an example of how we can use information to help the link estimator, we take a closer look at two collection protocols that are part of the TinyOS 2 distribution [3], the Collection Tree Protocol, or CTP [4], and MultiHopLQI [5]. CTP uses a probe-based link estimator, while MultiHopLQI relies solely on the link quality indicator (LQI) provided by the CC2420 radio chip. We have performed collection experiments with these protocols on an 85-node testbed at a low rate, with each node generating one packet every 10 seconds. Figure 2 shows a typical routing tree formed by CTP (a), MultiHopLQI (b), and a version of CTP with no restriction on the size of the link estimator tables (c). It also shows the average cost, in number of transmissions, for each delivered packet. Lower costs mean shorter paths with good quality.

CTP's cost is higher than MultiHopLQI's, even though the latter only uses physical layer information. This is the symptom of two problems. First, because CTP uses a bidirectional probe-based link estimator, its link table size limits a node's in-degree. Second, also because of the limited link table size, it may be that the best outgoing link is not even on the table to be selected for routing. Figure 2(c) shows that when the link table is unrestricted, CTP can outperform MultiHopLQI. In Section 4 we show how using information from the physical, link, and network layers we can completely mitigate these problems. The following subsections elaborate on the benefits and limitations of each layer.

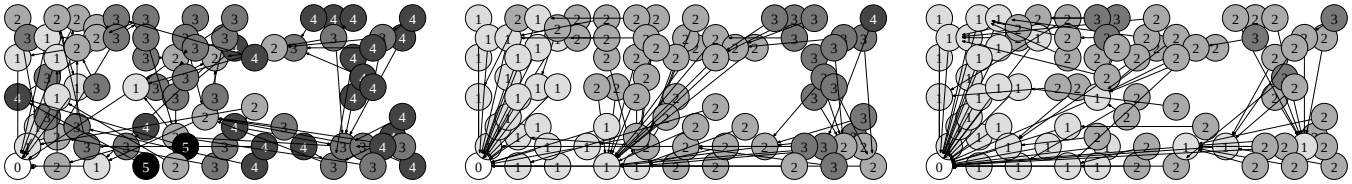
2.1 Physical Layer

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. In Figure 1, MultiHopLQI (c) and SP[1] (d) use physical layer information for link estimation.

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 [17], 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.

Figure 3 shows a limitation of physical layer information that we observed during a 12-hour low rate collection experiment using MultiHopLQI on a 94 node testbed. As Figure 1(c) shows, MultiHopLQI does not use link layer information. Although the protocol performed well overall, there were bursts of packet loss. As Figure 3 shows, for a period of time, the PRR between the nodes *C* and *P* dropped from an average of 0.9 to almost 0.6. This degradation in link quality was not accompanied by a drop in the decoding quality indicator (LQI). All of the packets *C* received had high quality: it just wasn't receiving all the packets.

[1] when the provided by the underlying radio



(a) CTP (cost = 3.14) (b) MultiHopLQI (cost = 2.28) (c) CTP unconstrained (cost = 1.86)

Figure 2. Routing trees formed on 85 node topology by CTP with 10-node link table, MultiHopLQI, and CTP with unrestricted link table. The average cost in transmissions per delivered message is in parenthesis. The root is the node in the bottom left corner, and darker nodes mean longer paths to the root.

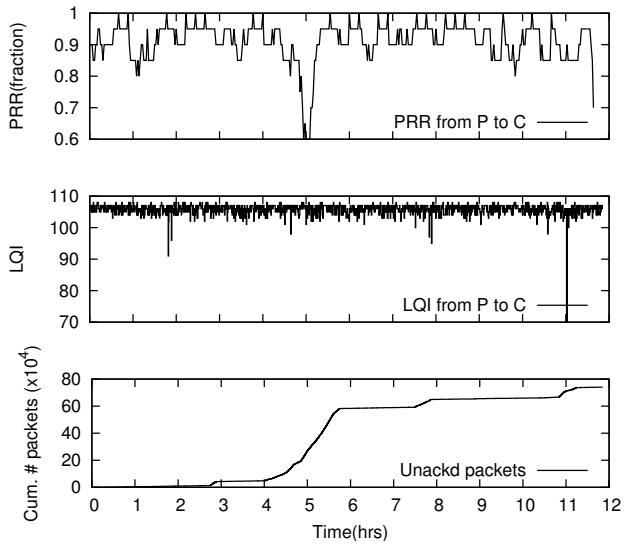


Figure 3. Unaware of the reduced PRR, MultiHopLQI attempts to deliver packets on the same link between the forth and sixth hour causing increased number of retransmissions due to unacknowledged packets.

2.2 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. In Figure 1(a) 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.

2.3 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 [10] benefits from neighbors that are evenly spread in all directions, while the S4 routing protocol [15] 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 (Figure 1(b)) integrates the link estimator into its routing layer. SP (Figure 1(d)) 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.

In the following section we describe how we can efficiently achieve cooperation between the link estimator and all three layers, with clean and well-defined interfaces using only four bits of information. We then demonstrate in Section 4 that indeed our interfaces allow significant performance gains through effective information exchange between the layers.

3 Design

Section 2 showed that each layer can measure or observe properties that aid link estimation. 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. This section proposes interfaces between the three layers and describes a link estimator that uses them.

3.1 Estimator interfaces

Figure 4 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

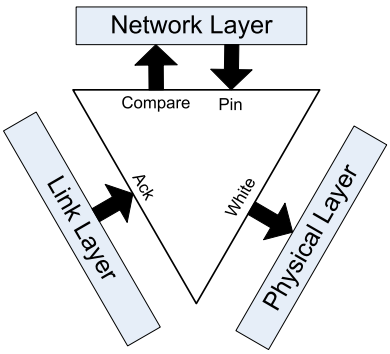


Figure 4. A link estimator, represented by the triangle in the center, uses four bits of information from the three layers. Outgoing arrows represent information the estimator requests on packets it receives. Incoming arrows represent information the layers actively provide.

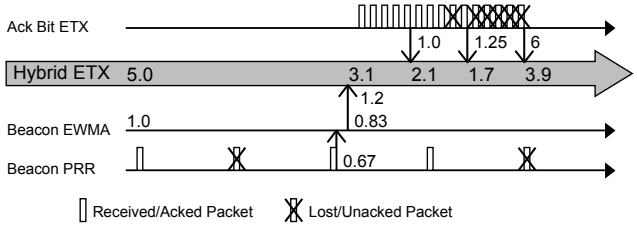


Figure 5. Our link estimator combines estimates of ETX separately for unicast and broadcast traffic with window sizes of $k_u = 5$ and $k_b = 2$ respectively. The latter are first themselves averaged before being combined. We show incoming packets are light boxes, marking dropped packets with an "×".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

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 3.3 below.

3.2 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.

3.3 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.

The estimator follows the basic table management algorithm outlined by Woo et al. [20], 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 [11]. We follow the link estimation method proposed by Woo et al. [20], 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

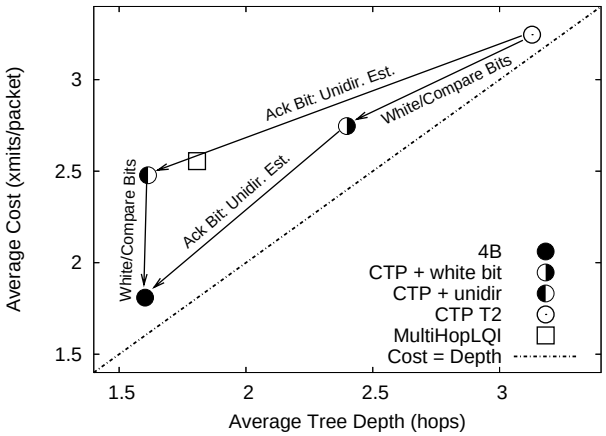


Figure 6. Exploring the link estimation design space: adding the ack bit and/or the white and compare bits to CTP decreases cost (lower is better) and the average depth of a node in the routing tree.

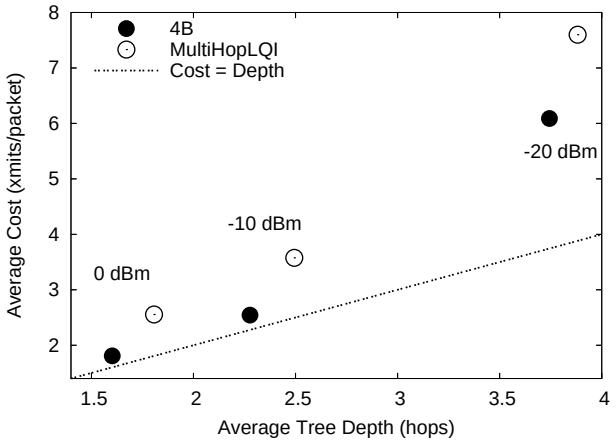


Figure 7. Average node depth and cost for MultiHopLQI and 4B for decreasing transmit powers on the Mirage testbed. 4B reduces cost by 19-28%.

values coming from the two estimators are combined in a second EWMA, as shown in Figure 5. 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 Evaluation

We have implemented a prototype of the link estimator described in Section 3.3 in TinyOS 2, and evaluate it by replacing the standard link estimator in CTP. Our estimator uses the four bit interfaces to the physical, link, and network layers, and we modified these layers in CTP to also use the interfaces. In this section we perform a detailed experimental comparison between our prototype, the original CTP, and MultiHopLQI. In the following discussion, we label our prototype as simply '4B'. As described above, MultihopLQI uses the Link Quality Indication (LQI), a feature of the CC2420 radio, and for that radio it is currently the best performing collection implementation for TinyOS.

In our comparison we run all three protocols on the Mirage testbed, using 85 MicaZ nodes with one node set as the basestation. We also ran experiments on a second testbed, TutorNet, using 94 TelosB nodes. 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 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 fact that the testbeds are static, and that all of our results agree from one testbed to the other gives us confidence in the results of the shorter runs.

The primary metric we use to evaluate performance 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. It takes into account the number of hops in a path, the number of per-link retransmissions needed, and also the wasted network effort in packets that are dropped. 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.

We first explore how the addition of each of the bits in Section 3.1 impacts cost and route length. We compare the original CTP with 4B, and two intermediary implementations. The uppermost-left point of Figure 6 shows the cost and depth of CTP running in the Mirage testbed. Adding unidirectional link estimation to CTP with the ack bit reduces average tree depth by 93%, and reduces cost by 31%. Unidirectional estimates decouple in-degree from the link table size, hence the large decrease in depth.

Adding the white and compare bits to the resulting protocol decreases cost to 55% of the original CTP, possibly because of improved parent selection. Adding only the white and compare bits to CTP provides 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 4B does better than MultiHopLQI. 4B has 29% lower cost and 11% shorter paths than MultiHopLQI. On another set of experiments on TutorNet, 4B's cost and average depth were respectively 44% and 9.7% lower than MultiHopLQI's. The trees produced by 4B are very similar qualitatively and in average depth to the trees produced by CTP with unrestricted link tables (Figure 2).

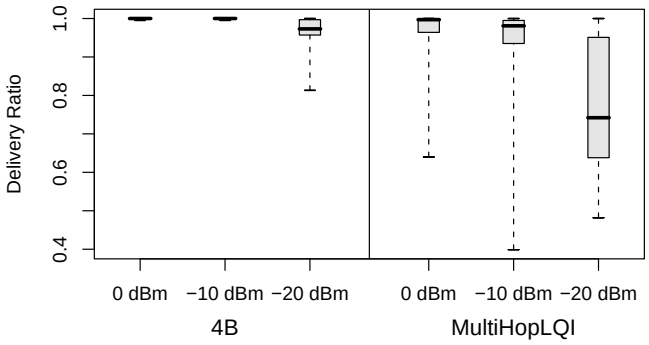


Figure 8. Boxplots of per-node delivery distributions at decreasing transmit power, for both MultihopLQI and 4B. Whiskers show the minimum and maximum values. Boxes show the 1st and 3rd quartiles. The line is the median. 4B maintains much higher and consistent delivery rates across the network.

Figure 7 compares the cost and average node depth of 4B and MultihopLQI in the Mirage testbed as transmit power varies from −20 dBm to 0 dBm. In each protocol, we see that both average node depth and cost increase with decreasing transmit power, as nodes need to route packets over more hops to get to the sink. 4B's improvement in cost over MultihopLQI ranges from 29% to 11%, and the improvement in average depth from 11% to 3.5%. 4B's cost, for the 0 and -10 dBm cases, is at most 13% above the lower bound, while it is at most 43.4% above the lower bound for MultiHopLQI. In a network with many hops, both protocols become less efficient. The relative increase in cost (62% above average depth for 4B and 95% for MultiHopLQI) are indicative of retransmissions and/or losses. Even with similar tree depths, however, 4B is able to select better links in this situation.

Figure 8 looks at the per-node distribution of delivery ratios for the same experiments, and gives some insight on why the costs in Figure 7 grow faster than the average depth. For 0 and 10 dBm, 4B showed an average delivery ratio above 99.9%, with minimum 99.3%. For 0 dBm, MultihopLQI's average delivery ratio over all nodes was 95.9%, with the worst node at 64%. As the transmit power decreases, the relative importance of RF noise increases, creating localized asymmetries in the network. As in the example of Section 2, MultiHopLQI's performance drops as some of this variation in link quality is not captured by the physical layer link quality indicator. We plan to look further into the dynamic behavior of the network, but the much smaller number of packet losses in 4B, even at -20dBm, indicates that most of the inefficiency seen in its cost is due to retransmissions, rather than loss. This suggests that the estimator is agile enough to notice packet losses and trigger the switch to a new route.

5 Conclusion

In this paper we have presented narrow, well-defined interfaces that allow a link estimator to use information from the physical, link, and network layers. Our prototype has shown significant improvements on cost and delivery ratio over the state of the art, while maintaining layered networking abstractions. This is encouraging, as we have not fully explored the possibilities of using our four bits of of interface. Looking forward, a portable, accurate, and efficient link estimator is ever more important with the growing popularity of PANs, and the extension of IP to low-power, embedded networks through efforts like IETF 6lowpan. For example, TCP's performance is known to be very sensitive to packet loss, and the improvements achieved using our link estimator may have a large impact on end-to-end throughput.

6 References

[1] 6lowpan charter. `http://www.ietf.org/html.charters/6lowpan-charter.html`.

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

[3] Tinyos 2. `http://www.tinyos.net/tinyos-2.x/doc`.

[4] Tinyos 2 tep 123: The collection tree protocol. `http://www.tinyos.net/tinyos-2.x/doc/txt/tep123.txt`.

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

[6] D. Aguayo, J. Bicket, S. Biswas, G. Judd, and R. Morris. Link-level measurements from an 802.11b mesh network. In *Proc. of the ACM SIGCOMM Conf.*, pages 121–132, Portland, OR, 2004.

[7] B. N. Chun, P. Buonadonna, A. AuYoung, C. Ng, D. C. Parkes, J. Shneidman, A. C. Snoeren, and A. Vahdat. Mirage: A Microeconomic Resource Allocation System for SensorNet Testbeds. In *Proc. of the 2nd IEEE Workshop on Embedded Networked Sensors (EmNets 2005)*, May 2005.

[8] D. D. Couto, D. Aguayo, J. Bicket, and R. Morris. A High-Throughput Path Metric for Multi-Hop Wireless Routing. In *Proc. of the ACM MobiCom Conf.*, San Diego, CA, Sept. 2003.

[9] O. Gnawali, K.-Y. Jang, J. Paek, M. Vieira, R. Govindan, B. Greenstein, A. Joki, D. Estrin, and E. Kohler. The tenet architecture for tiered sensor networks. In *Proc. of the ACM SenSys Conf.*, pages 153–166, Boulder, CO, Nov. 2006.

[10] B. Karp and H. T. Kung. GPSR: greedy perimeter stateless routing for wireless networks. In *International Conference on Mobile Computing and Networking (MobiCom 2000)*, pages 243–254, Boston, MA, USA, 2000.

[11] K.-H. Kim and K. Shin. On Accurate Measurement of Link Quality in Multi-hop Wireless Mesh Networks. In *Proc. of the ACM MobiCom Conf.*, pages 38–49, Los Angeles, CA, Sept. 2006.

[12] K. Langendoen, A. Baggio, and O. Visser. Murphy Loves Potatoes: Experiences from a Pilot Sensor Network Deployment in Precision Agriculture. In *IEEE Parallel and Distributed Processing Symposium*, Rhodes Island, Greece, Apr. 2006.

[13] R. Mahajan, M. Rodrig, D. Wetherall, and J. Zahorjan. Analyzing the MAC-level behavior of Wireless Networks in the Wild. In *Proc. of the ACM SIGCOMM Conf.*, pages 75–86, Pisa, Italy, Aug. 2006.

[14] G. Mainland, M. Welsh, and G. Morrisett. Flask: A Language for Data-driven Sensor Network Programs. Technical Report TR-13-06, Harvard Univ., 2006.

[15] Y. Mao, F. Wang, L. Qiu, S. Lam, and J. Smith. S4: Small State and Small Stretch Routing Protocol for Large Wireless Sensor Networks. In *Proc. of the USENIX NSDI Conf.*, Cambridge, MA, Apr. 2007.

[16] S. Rangwala, R. Gummadi, R. Govindan, and K. Psounis. Interference-Aware Fair Rate Control in Wireless Sensor Networks. In *Proc. of the ACM SIGCOMM Conf.*, pages 63–74, Pisa, Italy, Aug. 2006.

[17] K. Srinivasan, P. Dutta, A. Tavakoli, and P. Levis. Some Implications of Low Power Wireless to IP Networking. In *Proc. of the ACM HotNets Conf.*, pages 31–37, Irvine, CA, 2006.

[18] J. P. Vasseur. In *IETF 69th Meeting*.

[19] G. Werner-Allen, P. Swieskowski, and M. Welsh. Fidelity and Yield in a Volcano Monitoring Sensor Network. In *Proc. of the ACM OSDI Conf.*, Seattle, WA, Nov. 2006.

[20] A. Woo, T. Tong, and D. Culler. Taming the Underlying Challenges of Reliable Multihop Routing in Sensor Networks. In *Proc. of the ACM SenSys Conf.*, pages 14–27, Los Angeles, CA, Nov. 2003.

[21] J. Zhao and R. Govindan. Understanding Packet Delivery Performance in Dense Wireless Sensor Networks. In *Proc. of the ACM SenSys Conf.*, pages 1–13, Los Angeles, CA, Nov. 2003.

[22] M. Zuniga and B. Krishnamachari. An Analysis of Unreliability and Asymmetry in Low-Power Wireless Links. *Transactions on Sensor Networks*, 3(2), 2007.