A High-Throughput Path Metric for Multi-Hop Wireless Routing

Douglas S. J. De Couto   Daniel Aguayo   John Bicket   Robert Morris

M.I.T. Computer Science and Artificial Intelligence Laboratory

{decouto, aguayo, jbicket, rtm}@csail.mit.edu

http://www.pdos.lcs.mit.edu/grid

Abstract

This paper presents the *expected transmission count* metric (ETX), which finds high-throughput paths on multi-hop wireless networks. ETX minimizes the expected total number of packet transmissions (including retransmissions) required to successfully deliver a packet to the ultimate destination. The ETX metric incorporates the effects of link loss ratios, asymmetry in the loss ratios between the two directions of each link, and interference among the successive links of a path. In contrast, the minimum hop-count metric chooses arbitrarily among the different paths of the same minimum length, regardless of the often large differences in throughput among those paths, and ignoring the possibility that a longer path might offer higher throughput.

This paper describes the design and implementation of ETX as a metric for the DSDV and DSR routing protocols, as well as modifications to DSDV and DSR which allow them to use ETX. Measurements taken from a 29-node 802.11b test-bed demonstrate the poor performance of minimum hop-count, illustrate the causes of that poor performance, and confirm that ETX improves performance. For long paths the throughput improvement is often a factor of two or more, suggesting that ETX will become more useful as networks grow larger and paths become longer.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Wireless communication*; C.2.2 [**Computer-Communication Networks**]: Network Protocols—*Routing protocols*

General Terms

Design, Experimentation, Measurement, Performance

Keywords

Multi-hop wireless networks, Ad hoc networks, Rooftop networks, Wireless routing, Route metrics, 802.11b, DSR, DSDV, ETX

This research was supported by grants from NTT Corporation under the NTT-MIT collaboration, and by MIT's Project Oxygen.

MobiCom '03, September 14–19, 2003, San Diego, California, USA.

1. Introduction

Much of the recent work in ad hoc routing protocols for wireless networks [25, 15, 26] has focused on coping with mobile nodes, rapidly changing topologies, and scalability. Less attention has been paid to finding high-quality paths in the face of lossy wireless links. This paper presents measurements of link loss characteristics on a 29-node 802.11b test-bed, and uses these measurements to motivate the design of a new metric which accounts for lossy links: *expected transmission count* (ETX).

The metric most commonly used by existing ad hoc routing protocols is minimum hop-count. These protocols typically use only links that deliver routing probe packets (query packets, as in DSR or AODV, or routing updates, as in DSDV). This approach implicitly assumes that links either work well or don't work at all. While often true in wired networks, this is not a reasonable approximation in the wireless case: many wireless links have intermediate loss ratios. A link that delivers only 50% of packets may not be useful for data, but might deliver enough routing update or query packets that the routing protocol uses it anyway.

Minimizing the hop-count maximizes the distance traveled by each hop, which is likely to minimize signal strength and maximize the loss ratio. Even if the best route is a minimum hop-count route, in a dense network there may be many routes of the same minimum length, with widely varying qualities; the arbitrary choice made by most minimum hop-count metrics is not likely to select the best. One contribution of this paper is to quantify these effects (Section 2).

One approach to fixing this problem is to mask transmission errors. For example, the 802.11b ACK mechanism resends lost packets, making all but the worst 802.11b links appear loss-free. However, retransmission does not make lossy links desirable for use in paths: the retransmissions reduce path throughput and interfere with other traffic. Another approach might be to augment minimum hop-count routing with a threshold that ignores lossy links, but a lossy link may be the only way to reach a certain node, and there might be significant loss ratio differences even among the above-threshold links.

The solution proposed and evaluated in this paper is the ETX metric. ETX finds paths with the fewest expected number of transmissions (including retransmissions) required to deliver a packet all the way to its destination. The metric predicts the number of retransmissions required using per-link measurements of packet loss ratios in both directions of each wireless link. The primary goal of the ETX design is to find paths with high throughput, despite losses.

In order to demonstrate that ETX is effective, this paper presents measurements taken from the test-bed network. These measure-

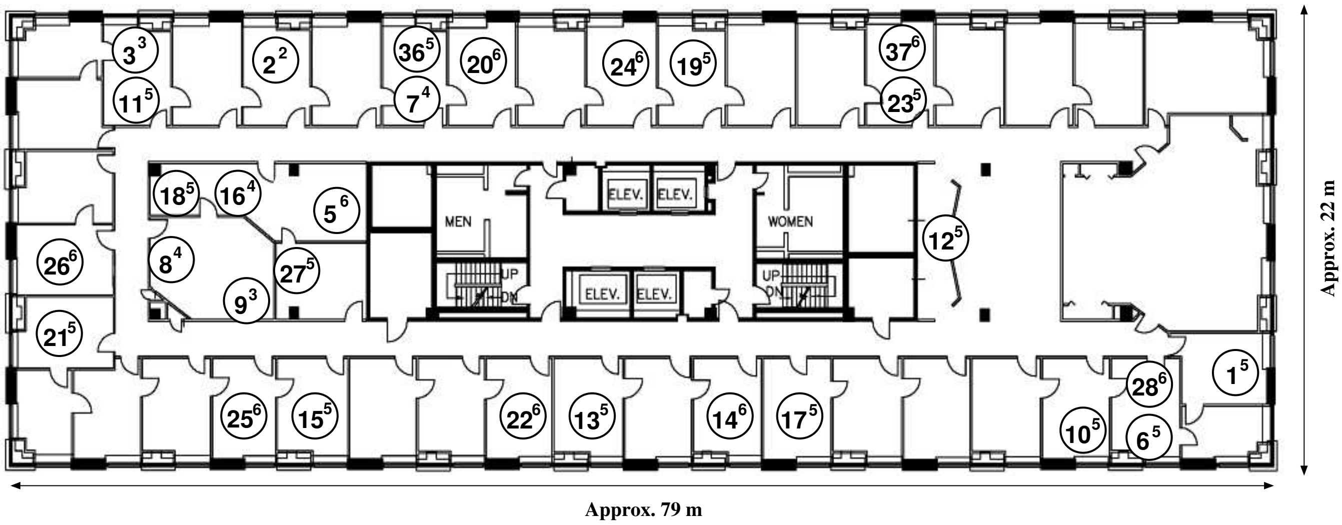

Figure 1: A map of the test-bed. Each circle is a node; the large number is the node ID, and the superscript indicates which floor of the building the node is on.

ments show that ETX improves the throughput of multi-hop routes by up to a factor of two over a minimum hop-count metric. ETX provides the most improvement for paths with two or more hops, suggesting that transmission count offers increased benefit as networks grow larger and paths become longer.

This paper makes the following main contributions. First, it explores the details of the performance of minimum hop-count routing on a wireless test-bed, and explains why minimum hop-count often finds routes with significantly less throughput than the best available. Second, it presents the design, implementation, and evaluation of the ETX metric. Third, it describes a set of detailed design changes to the DSDV [25] and DSR [15] protocols (to which ETX is an extension), that enable them to more accurately choose routes with the best metric.

This work is part of an effort to deploy a production-quality multi-hop rooftop 802.11b network. The initial version of that network was almost unusable due to the effects detailed in Section 2. The larger goal of this work is to help make such networks a practical reality.

The paper proceeds in Section 2 with an analysis of the problems with minimum hop-count routing. Section 3 describes the design of the new ETX metric, and Section 4 describes how ETX is implemented, including changes to DSDV and DSR. Section 5 evaluates ETX using experiments on the test-bed. Section 6 describes related work, and Section 7 concludes the paper.

2. Performance of Minimum-Hop-Count Routing

This section shows that minimum hop-count routing typically finds routes with significantly lower throughput than the best available. The evidence comes from measurements of DSDV on a test-bed network. We explain why minimum hop-count does poorly by looking at the distribution of route throughputs and link loss ratios.

2.1 Experimental Test-Bed

All the data in this paper are the result of measurements taken on a 29-node wireless test-bed. Each node consists of a stationary Linux PC with a Cisco/Aironet 340 PCI 802.11b card and an omni-directional 2.2 dBi dipole antenna (a "rubber duck"). The nodes are placed in offices on five consecutive floors of an office building. Their positions are shown in Figure 1.

The 802.11b cards are configured to send at one megabit per second (Mbps) with one milliwatt (mW) of transmit power. RTS/CTS is turned off, and the cards are set to "ad hoc" (IBSS, DCF) mode. Each data packet in the following measurements consists of 24 bytes of 802.11b preamble, 31 bytes of 802.11b and Ethernet encapsulation header, 134 bytes of data payload, and 4 bytes of frame check sequence: 193 bytes in total. An 802.11b ACK packet takes 304 microseconds to transmit, the inter-frame gap is 60 microseconds, and the minimum expected mandatory back-off time is 310 microseconds, resulting in a total time of 2,218 microseconds per data packet. This gives a maximum throughput of 451 unicast packets per second over a loss-free link.

While the test-bed itself carried only the data and control traffic involved in each experiment, interference of various kinds was inevitably present. In particular, each floor of the building has four 802.11b access points, on various different channels.

The DSDV implementation used in this paper is new, with modifications described in Section 4.

2.2 Path Throughputs

Figure 2 compares the throughput of routes found with a minimum hop-count metric to the throughput of the best routes that could be found. Each curve shows the throughput CDF (in packets per second) for 100 node pairs; the pairs are randomly selected from the $29 \times 28 = 812$ total ordered pairs in the test-bed. A point's x value indicates throughput, in packets per second; the y value indicates what fraction of pairs had less throughput. The left curve is the throughput CDF achieved by routing data using DSDV with the minimum hop-count metric. The right curve is the throughput CDF for the best known route between each pair of nodes. Packets were only sent between one pair at a time. For each pair, the DSDV and best-path tests were run immediately after one another, to limit variation in link conditions over time.

The "best" path between each pair of nodes was found by sending data along ten potential best paths, one at a time, and selecting the path with the highest throughput. Potential best paths were identified by running an off-line routing algorithm, using as input measurements of per-link loss ratios, similar to those in Section 2.4,

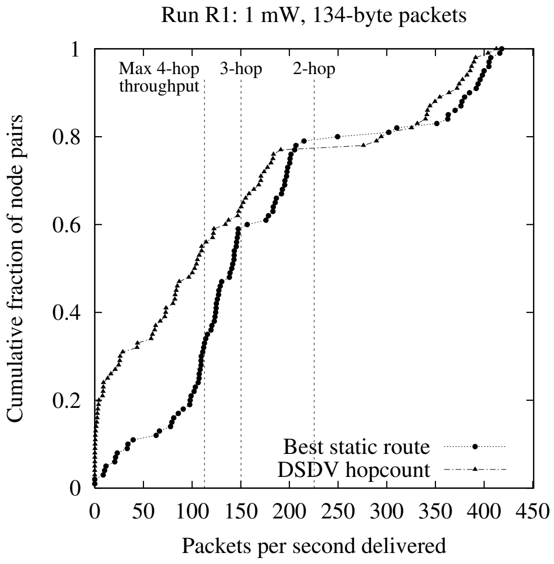

Figure 2: When using the minimum hop-count metric, DSDV chooses paths with far less throughput than the best available routes. Each line is a throughput CDF for the same 100 randomly selected node pairs. The left curve is the throughput CDF of DSDV with minimum hop-count. The right curve is the CDF of the best throughput between each pair, found by trying a number of promising paths. The dotted vertical lines mark the theoretical maximum throughput of routes of each hop-count.

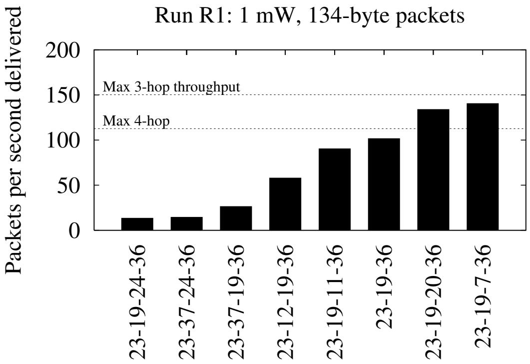

Figure 3: Throughput available between one pair of nodes, 23 and 36, along the best eight routes tested. The shortest of the routes does not perform the best, and there are a number of routes with the same number of hops that provide very different throughput.

and with a penalty to reflect the reduction in throughput caused by interference between successive hops of multi-hop paths. New link measurements were collected roughly every hour during the experiment; the best paths for each pair were generated using the most recently available loss data.

The values in Figure 2 are split into two main ranges, above and below 225 packets per second. The values above 225 correspond to pairs that communicated along single-hop paths; those at or below 225 correspond to multi-hop paths. A single-hop direct route can deliver up to about 450 packets per second, but the fastest two-hop route has only half that capacity. The halving is due to transmissions on the successive hops interfering with each other: the middle node cannot receive a packet from the first node at the same time it is sending a packet to the final node. Similar effects cause the fastest three-hop route to have a capacity of about $450/3 = 150$ packets per second.

Minimum hop-count performs well whenever the shortest route is also the fastest route, especially when there is a one-hop link with a low loss ratio. A one-hop link with a loss ratio of less than 50% will outperform any other route. This is the case for all the points in the right half of Figure 2. Note that the overhead of DSDV route advertisements reduces the maximum link capacity by about 15 to 25 packets per second, which is clearly visible in this part of the graph.

The left half of the graph shows what happens when minimum hop-count has a choice among a number of multi-hop routes. In these cases, the hop-count metric usually picks a route significantly slower than the best known. The most extreme cases are the points at the far left, in which minimum hop-count is getting a throughput close to zero, and the best known route has a throughput of about 100 packets per second. The minimum hop-count routes are slow because they include links with high loss ratios, which cause bandwidth to be consumed by retransmissions.

2.3 Distribution of Path Throughputs

Figure 3 illustrates a typical case in which minimum hop-count routing would not favor the highest-throughput route. The throughput of eight routes from node 23 to node 36 is shown. The routes are the eight best which were tested in the experiments described above.

The graph shows that the shortest path, a two-hop route through node 19, does not yield the highest throughput. The best route is three hops long, but there are a number of available three-hop routes which provide widely varying performance.

A routing protocol that selects randomly from the shortest hop-count routes is unlikely to make the best choice, particularly as the network grows and the number of possible paths between a given pair increases.

2.4 Distribution of Link Loss Ratios

Figure 4 helps explain why high-throughput paths are difficult to find. Each vertical bar corresponds to the direct radio link between a pair of nodes; the two ends of the bar mark the broadcast packet delivery ratio in the two directions between the nodes. To measure delivery ratios, each node took a turn sending a series of broadcast packets for five seconds, and counted the number of packets that the 802.11b hardware reported as transmitted. Packets contained 134 bytes of 802.11b data payload. Every other node recorded the number of packets received. The delivery ratio from node X to each node Y is calculated by dividing the number of packets received by Y by the number sent by X. The loss ratio of a link is one minus its delivery ratio. We use the term “ratio” instead of “rate” to avoid confusion with throughput delivery rates, which are expressed in packets per second.

Note that 802.11b broadcasts don’t involve acknowledgements or retransmissions. Because 802.11b retransmits lost unicast packets, the unicast packet loss ratio as seen by higher layers is far lower than the underlying loss ratio (depending on the maximum number of retransmissions allowed).

Three features of Figure 4 are important. First, a large fraction of the links have an intermediate delivery ratio in at least one direction. That is, they are likely to deliver some routing protocol

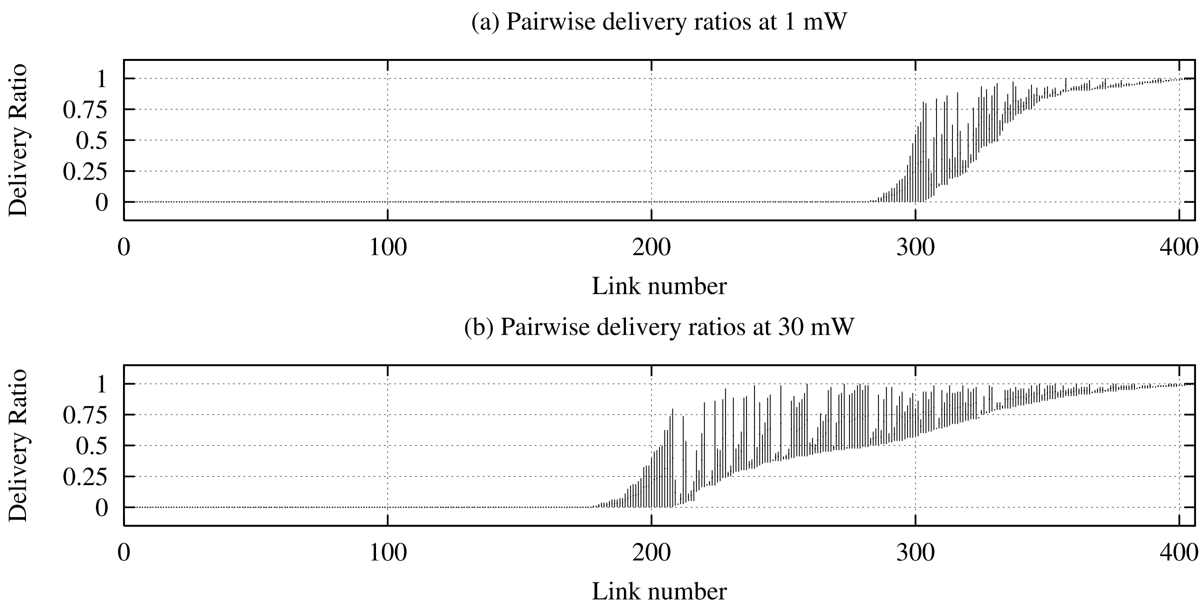

Figure 4: One-hop packet delivery ratios between each pair of hosts at 1 mW (above) and 30 mW (below). The top and bottom ends of each vertical line indicate the delivery ratios in the two directions; the bars in each graph are sorted by the minimum of the two directions, so the link numbers do not necessarily correspond. The packet size is 134 bytes of 802.11b data payload. Data for all 406 pairs of hosts are shown. Many links are asymmetric, and there is a wide range of loss ratios.

packets, but would lose many packets if used for data. Second, there is a full spectrum of link delivery ratios, so some advantage can be expected from making fine-grained choices between links when choosing paths. Third, many links have asymmetric delivery ratios.

Of the 406 node pairs in Figure 4a (1 mW), there are 124 with links which delivered packets in at least one direction. Of those links, 28 are asymmetric, with forward and reverse delivery ratios that differ by at least 25%. The 28 asymmetric links involve 22 different nodes. Because 802.11b uses link-level ACKs to confirm delivery, both directions of a link must work well in order to avoid retransmissions. Since most nodes in the network are involved in at least one asymmetric link, routing protocols must cope with asymmetry to be effective.

3. ETX Metric Design

This section describes the design of the ETX metric. The metric's overall goal is to choose routes with high end-to-end throughput. Section 2 suggests that the metric must account for the following issues:

- The wide range of link loss ratios.
- The existence of links with asymmetric loss ratios.
- The interference between successive hops of multi-hop paths.

A number of superficially attractive metrics are not suitable. Using hop-count as the metric while ignoring links with loss ratios above a certain threshold may cause some destinations to be unreachable. Using the product of the per-link delivery ratios as the path metric, in an attempt to maximize the end-to-end delivery probability, fails to account for inter-hop interference; this metric would view a perfect two-hop route as better than a one-hop route with a 10% loss ratio, when in fact the latter would have almost twice the throughput. The same objection applies to using the useful throughput of a path's bottleneck (highest-loss-ratio) link as the path's metric. ETX, however, addresses each of these concerns.

End-to-end delay is another potential metric, but changes with network load as interface queue lengths vary; this can cause routes to oscillate away from a good path once the path is used. Our goal is to design a metric that is independent of network load; load balancing can be performed with separate algorithms that use the information provided by ETX. We have implemented ETX as a metric for the DSDV and DSR routing protocols.

3.1 The Metric

The ETX of a link is the predicted number of data transmissions required to send a packet over that link, including retransmissions. The ETX of a route is the sum of the ETX for each link in the route. For example, the ETX of a three-hop route with perfect links is three; the ETX of a one-hop route with a 50% delivery ratio is two.

The ETX of a link is calculated using the forward and reverse delivery ratios of the link. The forward delivery ratio, d_f, is the measured probability that a data packet successfully arrives at the recipient; the reverse delivery ratio, d_r, is the probability that the ACK packet is successfully received. These delivery ratios can be measured as described below. The expected probability that a transmission is successfully received and acknowledged is $d_f \times d_r$. A sender will retransmit a packet that is not successfully acknowledged. Because each attempt to transmit a packet can be considered a Bernoulli trial, the expected number of transmissions is:

$$\text{ETX} = \frac{1}{d_f \times d_r} \tag{1}$$

ETX has several important characteristics:

- ETX is based on delivery ratios, which directly affect throughput.
- ETX detects and appropriately handles asymmetry by incorporating loss ratios in each direction.
- ETX can use precise link loss ratio measurements to make fine-grained decisions between routes.

- ETX penalizes routes with more hops, which have lower throughput due to interference between different hops of the same path [20].
- ETX tends to minimize spectrum use, which should maximize overall system capacity.

In addition, ETX may decrease the energy consumed per packet, as each transmission or retransmission may increase a node's energy consumption.

The delivery ratios d_f and d_r are measured using dedicated link probe packets. Each node broadcasts link probes of a fixed size, at an average period τ (one second in the implementation). To avoid accidental synchronization, τ is jittered by up to $\pm 0.1\tau$ per probe. Because the probes are broadcast, 802.11b does not acknowledge or retransmit them. Every node remembers the probes it receives during the last w seconds (ten seconds in our implementation), allowing it to calculate the delivery ratio from the sender at any time t as:

$$r(t) = \frac{\text{count}(t-w,t)}{w/\tau}$$

Count$(t-w,t)$ is the number of probes received during the window w, and w/τ is the number of probes that should have been received. In the case of the link $X \to Y$, this technique allows X to measure d_r, and Y to measure d_f. Because Y knows it should receive a probe from X every τ seconds, Y can correctly calculate the current loss ratio even if no probes arrive from X.

Calculation of a link's ETX requires both d_f and d_r. Each probe sent by a node X contains the number of probe packets received by X from each of its neighbors during the last w seconds. This allows each neighbor to calculate the d_f to X whenever it receives a probe from X.

The ETX of a route is the sum of the link metrics. DSR and DSDV accumulate the route metric as they forward updates and queries, respectively.

3.2 Discussion

ETX makes at least two assumptions about the link layer. First, ETX only makes sense for networks with link-layer retransmission, such as 802.11b. Second, ETX assumes that radios have a fixed transmit power level. With variable power radios, it might be preferable to maximize hop-count, thereby decreasing interference and minimizing the energy used by each packet [29, 12, 18].

ETX does not attempt to route around congested links, and thus should not suffer from the oscillations that sometimes plague load-adaptive routing metrics such as end-to-end delay. To a first approximation, the loss measurements used by ETX do not reflect how busy a link is; a busy link may cause a probe broadcast to be deferred, but won't ordinarily cause it to be lost. This won't always be true, however, since 802.11b broadcasts are vulnerable to collisions from hidden terminals, and the 802.11b MAC can be unfair under high load [4]. As a result, a node might never be able to send its probes, causing its neighbors to believe that the reverse delivery ratio had become zero.

If the highest-throughput path has three or fewer hops, ETX is likely to choose it: the throughput of such paths is determined by the total number of transmissions, since all of the hops interfere with each other [20]. If the best path has four or more hops, ETX may choose a slower path that has fewer hops, since the increased number of transmissions required by extra hops does not slow down throughput beyond three hops.

ETX does not specifically account for mobility. ETX may choose good paths despite mobility if the underlying routing protocol can propagate route metrics quickly enough, and if accurate link measurements are available. There is a tradeoff between the accuracy of link measurements and the protocol's responsiveness to mobility.

4. Implementation

The routing system in which ETX is implemented has four main parts: the Click toolkit [19], and Click-based implementations of DSDV, DSR, and the ETX link measurement algorithms. The implementations can run in user-space, but running in the kernel allows use of the priority queuing described below, as well as easy access to transmission failure notification from the 802.11 MAC layer.

The DSDV protocol is implemented following the description by Perkins and Bhagwat [25], with ambiguities resolved by consulting Broch et al. [5] and the Rice/CMU implementation in the *ns* simulator [1, 27]. The DSR implementation follows the IETF Internet-Draft, version 9 [16].

4.1 Operation of DSDV

DSDV is a distance-vector protocol, which uses sequence numbers to ensure freshness, and a *settling time* mechanism to avoid unnecessarily propagating routes with inferior metrics. We made four changes to the original DSDV design in order to ensure that it uses the path with the best known metric. Before describing those changes, we present an overview of how the published version of the protocol selects routes.

Every node has a routing table entry for each destination it knows about. This entry contains four fields: the destination's identifier (IP address), the next hop on the route to that destination, the latest sequence number heard for that destination, and the metric. A node forwards packets to the next hop specified by the current contents of its routing table.

Every node periodically broadcasts a route advertisement packet containing its complete routing table. This advertisement is known as a *full dump*, and occurs at the *full dump period*.

Each node maintains a sequence number for itself, which it increments and includes in its own entry in every full dump it originates. The node copies the sequence numbers for the other entries in the full dump from its routing table. The effect is that the sequence number field in a routing table entry or advertisement entry reflects the age of that entry's routing information.

When a node receives another node's route advertisement broadcast, it updates its own route entries as follows. Suppose node X receives an advertisement from Y for destination D with metric m and sequence number n. If n is newer than the sequence number in X's current entry for D, X replaces its current entry with the new route through Y. X also accepts the new route if the sequence number is the same, but m is better than the metric of the current route. If X has no route to D, it accepts the new route. Otherwise X ignores the advertised route.

Each route entry has an associated *weighted settling time* (WST). The settling time of a route entry with a given sequence number is the amount of time between when a route with the sequence number was first received, and the time when the best route with the same sequence number was received. The WST is the weighted average of the settling times for recent sequence numbers, and is updated whenever a route with a new sequence number is received.

The WST is used together with *triggered updates* to quickly propagate good routes through the network, while avoiding an explosion of broadcasts. Whenever a node replaces a route entry with a newly received entry, it propagates the new route to its neighbors by sending a triggered update which contains only the changed information. However, triggered updates are not sent until at least

$2\times$WST has passed since first hearing the current sequence number. This prevents nodes from advertising a new route which will likely be later replaced by a better route. In addition, regardless of each route entry's WST, triggered updates are sent at no more than a maximum specified rate. Triggered updates that are delayed are batched together and sent at the next available time.

Finally, DSDV specifies that triggered updates can become full dumps if a large enough fraction of the routes need a triggered update. In this case, all routes with an elapsed WST are included in the full dump, and the node's sequence number is incremented.

4.2 Changes to DSDV

The first change we made affects how the WST is used. The CMU *ns* DSDV implementation does not advertise a route until $2\times$WST has passed since that *particular* route was heard. However, according to our interpretation of the original DSDV description [25], the waiting time before advertising a route should start when the *first* route of each sequence number is heard.

The second change is that our implementation does not use link-level feedback (i.e. 802.11b transmission failures) to detect broken links and produce broken-route messages. Broch et al. [5] report that broken-route messages typically cause all routes to the particular destination to be broken throughout the whole network, not just those that use the broken link. Our implementation still generates broken-route messages when routing table entries time out, but this rarely occurs during the experiments presented in this paper.

The third change (called *no-dump*) is that full dumps are never sent on a triggered update, even if many routes have changed. Triggered updates contain only the changed routes, and full dumps are only sent at the full dump period.

The final change (called *delay-use*) is that a route is not used until it is allowed to be advertised. That is, a new route is not used until $2\times$WST has expired since its sequence number was first heard. With this change, the best route heard with the previous sequence number is used until the current sequence number's WST has expired. Unmodified DSDV always uses the latest route accepted for a given destination, even if it cannot yet advertise that route.

The purpose of the last change is to prevent DSDV from using routes with bad metrics. For example, if there is an asymmetric one-hop route, a node will always hear new sequence numbers along the one-hop link first. Without the change, DSDV is forced to use the new one-hop route for routing, even if the ETX metric is poor. In general, shorter routes deliver new sequence numbers first, causing the original DSDV to use shortest paths for some fraction of the time between successive sequence numbers, regardless of the metric in use. With this change, DSDV will use the best route with the previous sequence number until the WST has expired and the best route with the new sequence number has likely been heard.

Figure 5 shows pseudo-code for the DSDV routing table update and packet forwarding algorithms, including our changes.

For the experiments in this paper, the full dump period was 15 seconds, and routing table entries were timed out after 60 seconds. Triggered updates were issued at a maximum rate of one per second. All DSDV experiments used the three protocol changes described above, except for Section 5.1.2, which evaluates the delay-use modification.

The ETX implementation measures link loss ratios with small probe packets, as described in Section 3.1. Probes contain 134 bytes of 802.11b payload. An ETX node broadcasts one probe per second, and remembers probes received from neighbors over the last ten seconds. Using relatively small probes saves bandwidth; Section 5 shows that predictions based on small packets are still useful even when the data traffic consists of large packets.

```
handle_route_ad(Packet p) {
  foreach Route r in p do
    handle_update(r)
}

handle_update(Route r) {
  // curr[]: best route for current seq
  // old[]: best route for previous seq

  // add link-metric to r.metric
  update_metric(r);

  if (r.seq == curr[r.dest].seq
   && r.metric < curr[r.dest].metric) {
    curr[r.dest] = r;
    curr[r.dest].best_time = now;
    schedule_triggered_update(r);
  } else if (r.seq > curr[r.dest].seq) {
    // save best route of last seq no
    old[r.dest] = curr[r.dest];

    curr[r.dest] = r;
    curr[r.dest].first_time = now;
    curr[r.dest].best_time = now;

    // update settling time
    old_wst = old[r.dest].wst;
    best_t = old[r.dest].best_time;
    first_t = old[r.dest].first_time;
    curr[r.dest].wst = 0.88*old_wst +
      0.12*(best_t - first_t);

    schedule_triggered_update(r);
  }
  // ignore old seqnos and bad metrics
}

// returns next hop ip address for dst
lookup_route(IPAddress dst) {
  // use old route if we haven't yet
  // advertised current route
  if (curr[dst].first_time +
    2*curr[dst].wst > now)
    return old[dst].next_hop;
  else
    return curr[dst].next_hop;
}
```

Figure 5: DSDV pseudo-code, including the modifications described in Section 4.2. The WST parameters 0.12 and 0.88 are chosen to produce a reasonable average.

4.3 DSR Implementation

Our DSR implementation follows revision 9 of the IETF Internet-Draft specification [16], following the requirements for networks which require bidirectional links to send unicast data. The implementation is based on one developed by Audun Tornquist at the University of Colorado at Boulder [32]. This section reviews DSR's basic operation as described in the draft, and describes our modifications to support ETX.

DSR is a reactive routing protocol, in which a node issues a *route request* only when it has data to send. Route requests are flooded through the network, each node appending its own address to each request it receives, and then re-broadcasting it. Each new request includes a unique ID, which forwarders use to ensure they only forward each request once.

The request originator issues new requests for the same destination after an exponentially increasing back-off time. Route requests are issued with increasing time-to-live (TTL) values, to minimize the range and cost of flooding.

The destination issues a *route reply* in response to every forwarded request it receives. Each reply, which includes the route which was accumulated as the request was forwarded through the network, is source-routed back to the originator along the reverse route. The source node chooses a route using information from the route replies it receives, and source-routes data along this route.

Our implementation stores the results of route replies in a *link cache*, which stores information about each link separately. A node runs Dijkstra's shortest-path algorithm on its link cache to find the best route to a destination.

DSR uses feedback from the link layer to react to link failures. When the 802.11 card signals that no acknowledgment was received after the maximum number of retries, the forwarding node issues a *route error* back to the source, which removes the link from its link cache and then computes a new route. If the source cannot find a route using its link cache, it issues a new route request.

To deal with asymmetric links, each node maintains a *blacklist*, which lists immediate neighbors with unidirectional links to the the node. These are links over which the node might receive broadcast requests, but which are unsuitable for unicast traffic.

If a transmission failure occurs when forwarding a route reply, the neighbor to which the node was trying to forward the reply is added to the blacklist, with an entry of *unidirectionality probable*. From that point, the node will not forward route requests received over that link.

If the asymmetry of the link is not positively determined for some time, its entry is downgraded to *unidirectionality questionable*. If a route request is received over such a link, the node delays forwarding it while it issues a direct, one-hop unicast route request back to the questionable neighbor. If a reply is received, the node forwards the original route request and removes the blacklist entry. Otherwise, the node drops the request.

Entries are removed from the blacklist when the link is determined to be bidirectional, e.g. by a successful unicast transmission.

The DSR specification describes optimizations in which nodes update their link caches using data from packets they forward or "overhear". We did not implement any of the optimizations which require the wireless interface to operate in promiscuous receive mode. We also did not implement "reply from cache," in which forwarding nodes can respond to route requests with information from their own link caches. All link caches were flushed between experiments, so these decisions should not affect the results presented below.

The nodes do not perform packet salvage, in which forwarding nodes, in the event of a transmission failure or received route error, attempt to find alternate routes for queued packets. Each forwarding node queues only a few packets, so only a small number of packets are dropped in these cases.

To use the ETX metric, the implementation was modified in a few simple ways. Link probes are used to measure delivery ratios, as in the DSDV implementation. When a node forwards a request, it appends not only its own address, but also the metric for the link over which it received the request. These metrics are included in the route replies sent back to the sender. When a node receives a request which it has already forwarded, it forwards it again if the accumulated route metric is better than the best which it has already forwarded with this request ID. This increases the chances that the originator will hear about the route with the best metric.

Entries in the link cache are weighted by the metrics which were included in the route replies. The Dijkstra algorithm finds the route to the destination which has the minimum metric.

4.4 Router Configuration Details

If a node is sending large volumes of data, there is a danger that probe packets or routing protocol packets may be dropped or delayed due to a full queue. To mitigate this problem, the implementation maintains separate Click queues for data packets, protocol packets, and link probes. Each of these queues can hold five packets. These queues all drain into a single queue in the wireless adapter's memory, managed by the driver, which has a capacity of three packets. Loss-ratio probes enter the adapter's queue first, followed by protocol packets, then data packets.

The DSDV implementation looks up a packet's destination in the routing table after dequeuing the packet from the data queue, and just before handing the packet to the 802.11b card. This avoids committing to the next hop before queuing, and makes forwarding more responsive to changes in the routing table. This technique depends on the fact that the nodes have only one wireless interface.

The DSR implementation, on the other hand, adds the source-route header to data packets before inserting them into the queue. On a transmission failure or a received route error, a node removes and drops all enqueued packets which include the broken link in their source route. This ensures that the node experiencing the transmission failure does not spend additional time and spectrum retransmitting more packets over the broken hop.

5. Evaluation

This section presents experimental results that show that ETX often finds higher-throughput paths than minimum hop-count, particularly between distant nodes. It also explores the effects of a number of individual design decisions in the ETX algorithm.

Unless otherwise stated, the experimental setup is as follows. The test-bed, 802.11b configuration, and packet size are as described in Section 2.1. The DSDV implementation includes the improvements described in Section 4.2 for both ETX and the hop-count metric. The DSR implementation is as described in Section 4.3.

The data presented below were collected during a few separate "runs". An entire run takes about 30 hours. A run considers each pair of nodes in turn. For each pair, one "experiment" is run for each routing protocol variant. At the start of each experiment, the routing software is reset (all tables are cleared), then the routing protocol and/or ETX probe algorithm is allowed to run long enough to stabilize (typically 90 seconds). Then the sending node of the pair sends data packets as fast as 802.11b allows it through the routing system to the destination. The destination measures the rate at which packets arrive. This arrangement ensures that the results from different protocols for the same pair of nodes are comparable, since the relevant experiments are run within a few minutes of each other.

Each graph below is labeled with the run from which it came. Graphs with the same run number are comparable. Graphs with different run numbers should not be compared, since the network's behavior changes substantially with time. The graphs below do not include error bars, but are representative of the many runs we have performed.

In DSDV experiments using ETX or minimum hop-count, the routing protocol runs for 90 seconds, immediately after which the source sends data packets as fast as possible for 30 seconds. As described in Section 3.2, the heavy load causes the MAC protocol to become extremely unfair, distorting the ETX measurements. To minimize the effects of MAC unfairness, every node routes packets using a snapshot of its route table taken at the end of the 90-second warm-up period, before any data is sent.

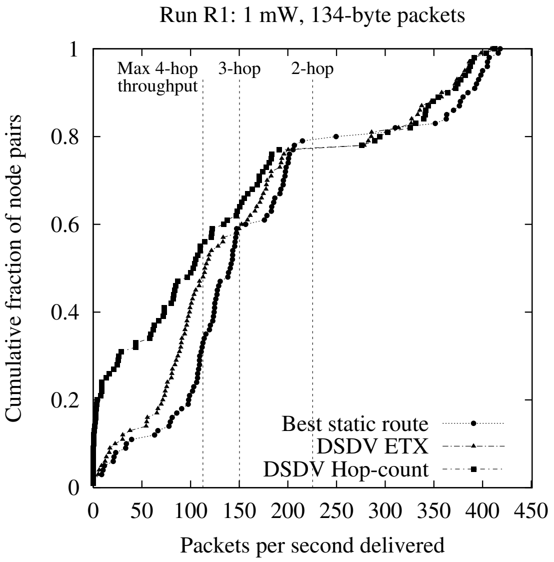

Figure 6: ETX finds higher throughput routes than minimum hop-count. This data is taken from the same experimental run as Figure 2. Each point represents one of 100 node pairs.

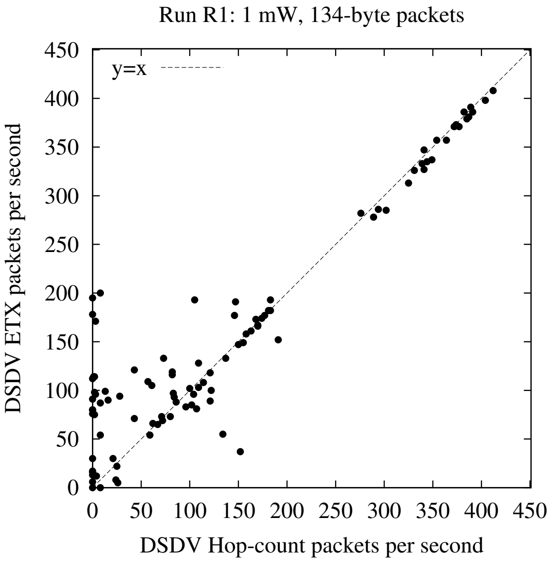

Figure 7: The ETX and hop-count data from Figure 6, plotted on a per-pair basis. The x value of each point shows that pair's throughput for DSDV with minimum hop-count; the y value shows the throughput for DSDV with ETX. Points above the line $y = x$ are pairs where ETX outperformed hop-count.

In DSR experiments with ETX or minimum hop-count, a source starts by sending one data packet per second for five seconds. This ensure that DSR finds a route before throughput measurements are taken. After the five seconds passes, the source sends packets as fast as possible for 30 seconds. In DSR experiments with ETX, the source waits an additional 15 seconds before initiating the route request, to give the nodes time to accumulate link measurements.

All experiments run with the appropriate routing overhead. That is, while measuring the throughput of routing with the ETX metric, nodes send periodic ETX broadcast probes. While measuring the throughput of DSDV (with either metric), nodes sends DSDV routing advertisements, just as a production routing system would.

5.1 Metric Performance with DSDV

Figure 6 compares the throughput CDFs of paths found by DSDV using ETX and minimum hop-count, between 100 randomly chosen node pairs. This data is taken from the same run as in Figure 2, and shows that DSDV using the ETX metric often finds much faster routes than the minimum hop-count metric.

There are two main regions in Figure 6. The right half shows node pairs that could communicate directly, with loss ratios less than about 50% (i.e. with throughput greater than the maximum possible two-hop throughput of 225 packets per second). In these cases the minimum hop-count metric finds the one-hop route, which is the best route, and there is no opportunity for ETX to perform better. The left half corresponds to node pairs with a high direct loss ratio, for which the best route has more than one hop. In this region, the sensitivity of ETX to differences among the many different paths of the same length allows it often to find better paths than hop-count.

Figure 7 shows the same data as Figure 6, but organized as a scatter plot to allow a direct comparison between the performance of each metric for individual pairs. Each pair is represented by one point; the point's y value is the throughput obtained by DSDV using ETX, and the x value is the throughput obtained by DSDV using minimum hop-count. The upper-right quadrant shows pairs where ETX and minimum hop-count both used the one-hop path.

ETX outperforms minimum hop-count by the greatest margin when the hop-count metric uses links with very asymmetric loss ratios. This is illustrated by the points with x near zero and with y relatively large. Minimum hop-count is using links that deliver routing updates in one direction but deliver few or no data packets in the other, while ETX correctly avoids those links.

The points for two pairs in Figure 7 lie well below the $y = x$ line; this is because of variations in link quality between the ETX and minimum hop-count tests for those pairs. For the first pair, both ETX and hop-count used the same route, so the difference is due to an underlying change in the route's throughput. For the second pair, ETX used a slower 3-hop path while hop-count used a two-hop path; ETX avoided using one of the links in the two-hop path because the measured delivery ratios were very poor. It is likely that the link's quality was different for the ETX and hop-count tests.

ETX incurs more overhead than minimum hop-count, due to its loss-ratio probes, but this overhead is small compared to the gains in throughput that ETX provides. ETX found usable routes for many pairs where minimum hop-count was delivering essentially zero packets per second.

Figure 8 shows the throughput for packets with a 1,386-byte payload. Although ETX still offers an improvement over minimum hop-count, the gain is not as large as for small packets. This is because ETX is still using small probes to estimate the link metrics. Since small packets are more likely to be delivered, ETX is incorrectly over-estimating the quality of each link and causing DSDV to pick sub-optimal routes. For example, if the single-hop direct route between two nodes has an ETX probe delivery rate of 51%, ETX will use it; however, the delivery rate of 1,386-byte packets on such a link is likely to be closer to 1%, so a route with more but higher-quality links would have been preferable. However, the small packets are still useful for detecting very asymmetric links, which is why ETX's gain over minimum is more pronounced to the left of the graph, where hop-count used very asymmetric links.

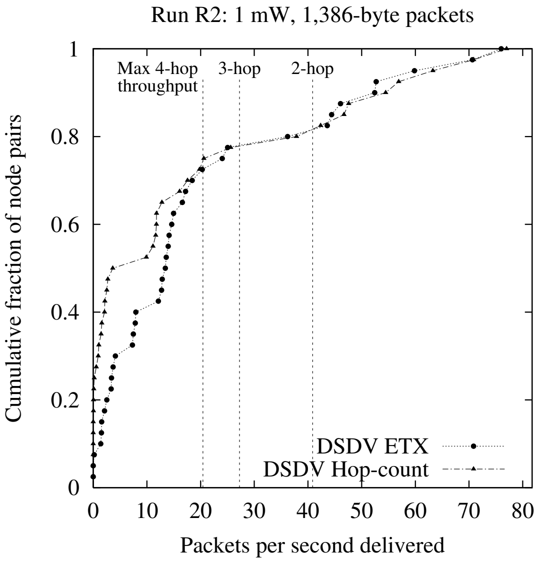

Figure 8: ETX provides less of a throughput advantage over minimum hop-count when using large (1,386-byte) packets. The small packets used to measure link loss ratios incorrectly predict the actual transmission counts for large packets. This graph shows 40 pairs randomly chosen from the 100 pairs used in the previous figures. The maximum 1-hop throughput of 1,386-byte data packets sent at 1 Mbps is 82 packets per second.

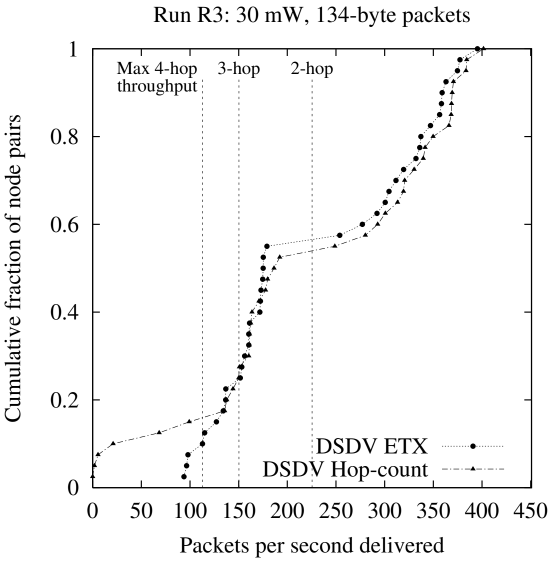

Figure 9: ETX versus minimum hop-count when transmitting at 30 mW, for 40 pairs. Using a higher transmit power produces a more highly connected network with many more links and a lower average hop-count, but ETX still provides some advantage.

Figure 9 shows the results of ETX versus minimum hop-count from a third run with the radios transmitting at 30 mW instead of 1 mW. The packet size is 134 bytes. When nodes send at the higher transmit power they have more links, as shown in Figure 4. This makes the network much more connected, decreasing the average hop-count required for nodes to communicate. As a result, ETX has fewer routes to choose from, and minimum hop-count has a lower chance of choosing a bad route. Figure 9 shows that ETX still provides some advantage in the more highly connected network.

5.1.1 *Impact of Asymmetry*

Some fraction of ETX's gains comes from avoiding extremely asymmetric links. The problem of routing when there are asymmetric links has been addressed in previous work by Lundgren et al. [22] and by Chin et al. [8]. These authors propose a link handshaking scheme to detect and avoid asymmetric links. In this scheme, a node X only accepts route updates from a neighboring node Y if Y is advertising a direct route to X. A node bootstraps the handshake by advertising provisional route entries, which indicate that the node has 'seen' another node, but not yet accepted routes from it.

We implemented the handshaking scheme for DSDV with the minimum hop-count metric. Figure 10 compares link handshaking to the ETX and minimum-hop-count metrics. Although link handshaking often improves over minimum hop-count alone, ETX finds faster routes. ETX's link measurements allow ETX to discriminate between links with varying degrees of asymmetry.

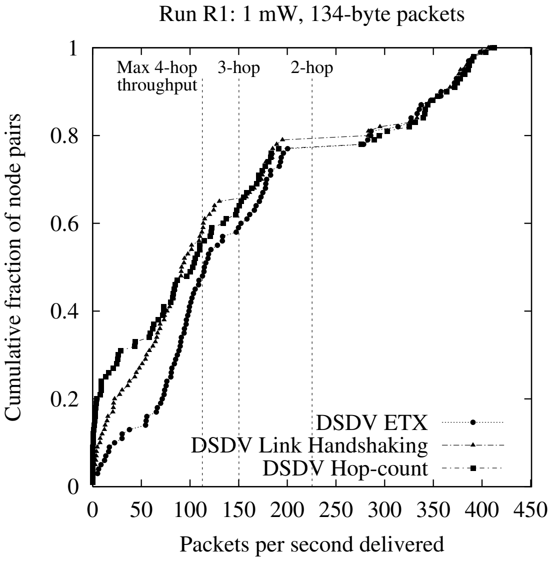

Figure 10: ETX provides a significant throughput over a simple handshaking scheme which avoids very asymmetric routes. ETX can make fine-grained decisions between links with varying degrees of asymmetry.

5.1.2 *Effects of DSDV Modifications*

Section 4.2 described modifications to DSDV designed to increase its responsiveness to metrics. The *delay-use* modification causes DSDV to delay using a newly received route until it is permitted to advertise the route (i.e. $2\times$WST has passed). Figure 11 shows that the delay-use modification improves the performance of DSDV with ETX.

5.2 Metric Performance with DSR

This section evaluates the performance of the DSR routing protocol with the ETX metric. As described in Section 4.3, DSR uses link-layer transmission failure feedback to help it avoid bad routes.

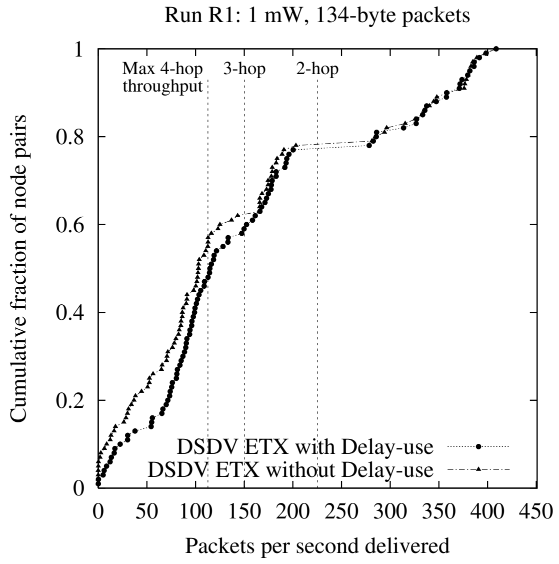

Figure 11: DSDV ETX with and without the delay-use modification to DSDV. This modification helps DSDV obey the link metric.

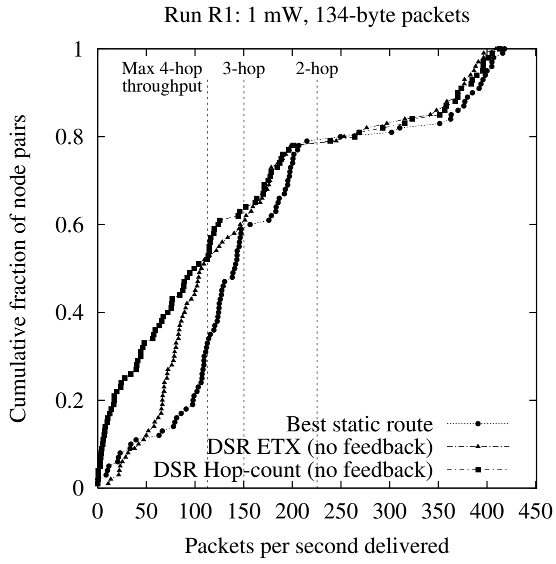

Figure 12: Throughput CDFs for DSR ETX compared with DSR hop-count, with link-layer transmission feedback disabled. ETX significantly improves initial route selection.

To isolate the effects of using ETX with DSR, we evaluated DSR performance both with and without link-layer feedback enabled.

Figure 12 shows the effect of using the ETX metric with DSR without link-layer feedback, for the same 100 pairs as in Figure 2. Because DSR never learns about transmission failures, no forwarding node ever issues any route errors. Thus DSR uses only the best route found by the initial route request, as determined by the metric.

Figure 12 shows that ETX greatly improves initial route selection in DSR compared to minimum hop-count. This is consistent with the DSDV results in Section 5.1. Minimum hop-count essentially chooses randomly from all the shortest routes the source obtains from the initial route request; as illustrated in Figure 3, this is often not the best route. ETX helps the source picks an initial route with high throughput.

Figure 13 illustrates the performance of ETX with DSR's link-layer feedback enabled. ETX provides a small benefit to some pairs in the intermediate and low throughput ranges (the middle and bottom of the CDF). However, failure feedback alone allows DSR to perform almost as well as DSR with ETX.

5.3 Accuracy of Link Measurements

For all the experiments in this paper, ETX used 134-byte packets to estimate link loss ratios. However, the loss ratios experienced by data packets of other sizes are likely to differ from the ETX estimate. Figure 14 shows how loss ratios vary with packet size for a number of different links. The experiments for these data were virtually identical to the broadcast delivery tests described in Section 2.4, except the broadcasting nodes sent packets of eighteen different sizes, from 50 to 1500 bytes. Seven sets of these experiments were conducted over the course of two days, and the results were averaged to minimize short-term variation. The x values represent packet size, and the y values show the delivery ratio for broadcast packets of that size. The lines connect the data points for the same host pairs.

The figure shows that packet size has a significant effect on delivery ratios. ETX, however, uses a single packet size to estimate link metrics. This is likely to lead to inaccurate metrics for packet sizes other than the size used for measurement probes. Furthermore, ETX uses the single packet size measurements to estimate the delivery ratio of link-layer ACK packets. Since 802.11b ACK packets are smaller than any 802.11b data packet, ETX always underestimates the delivery ratio of ACK packets. ACK packets are only 38 bytes in total, including all 802.11b overhead, while the 134-byte data packets used in most of the experiments are 193 bytes with 802.11b overhead.

Figure 15 shows the accuracy of ETX's transmission count predictions. Each point on the graph represents an experiment on a link between two randomly selected nodes. For each experiment, the two nodes first take turns sending broadcast probes with a 104-byte data payload. Each turn consists of ten probe broadcasts over the course of one second. After sixty seconds (thirty turns for each node), one of the nodes sends unicast traffic to the other for ten seconds, as fast as 802.11b allows. Each node logs packets sent and received. For each sent packet—that is, packets which the wireless interface actually attempted to transmit—the node asks the 802.11b hardware whether the transmission succeeded, and how many times the hardware re-transmitted the packet. The y values on the graph represent the average actual transmission counts. The x values are the expected transmission count based on the broadcast delivery ratios in the preceding minute, as calculated by Equation 1. Figure 15 shows that ETX tends to overestimate the number of required transmissions, probably because it underestimates the ACK delivery ratio for each link.

6. Related Work

The behavior of routing protocols over lossy links has been addressed and evaluated by real implementations in several papers. Lundgren et al. [22] coin the term "gray zones" to refer to links that deliver routing protocol data but not data traffic. They propose using link handshaking and counting route broadcasts to filter out gray zone links. Chin et al. [8] also propose link handshaking to filter out asymmetric links. Hu and Johnson [14] describe how to preemptively issue DSR route requests, based on link SNR values.

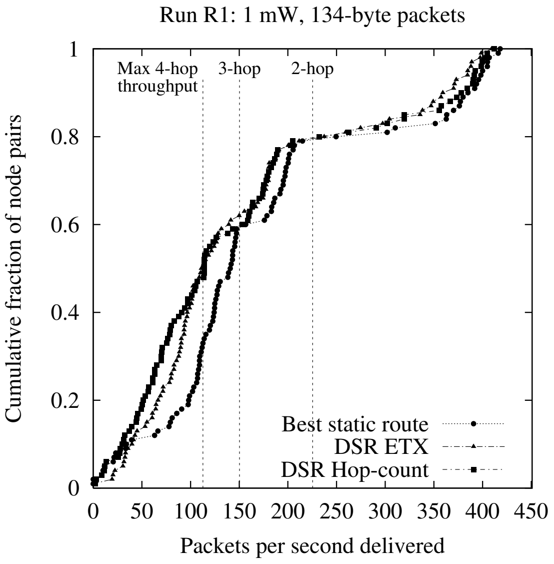

Figure 13: DSR ETX compared with DSR hop-count, with link-layer transmission feedback enabled. ETX only slightly improves overall DSR performance, because link-layer transmission failure feedback already helps DSR avoid links with high loss ratios.

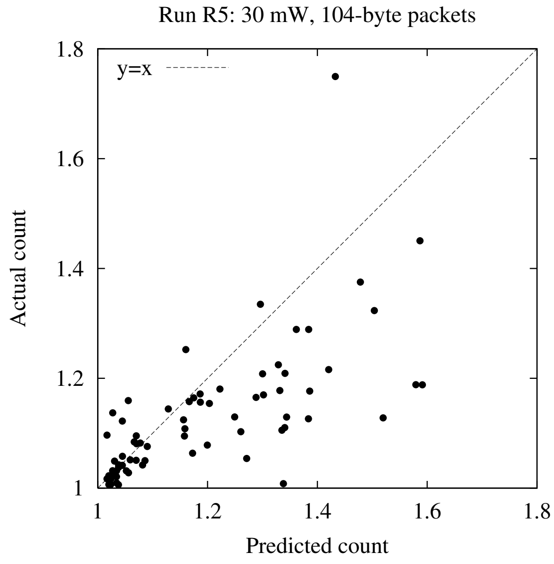

Figure 15: Because ETX measures with 134-byte probes, it underestimates ACK delivery ratios and therefore overestimates the total number of transmissions per packet. The graph shows the average actual transmission count for delivered unicast packets, versus a prediction based on bidirectional delivery ratios of broadcast probes in the preceding minute.

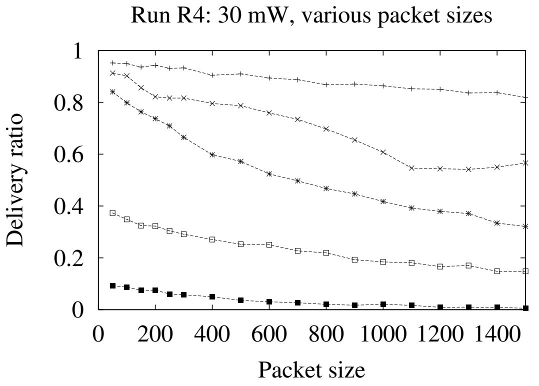

Figure 14: Delivery ratio decreases for larger packets. The graph shows the broadcast delivery ratio versus packet size, for a few links on the test-bed. The links shown are, from top to bottom: $12 \rightarrow 1$, $11 \rightarrow 13$, $23 \rightarrow 36$, $21 \rightarrow 19$, **and** $17 \rightarrow 18$.

Yarvis et al. [33] observe that hop-count performs poorly as a routing metric for a sensor network, and present the results of using a loss-aware metric instead. Their path metric approximates the product of the per-link delivery ratios. As argued in Section 3, this metric is likely to use low-loss paths with many hops in situations where a path with a smaller number of higher loss links would perform better.

Awerbuch et al. [2] present a metric to help find high-throughput paths when different links can run at different bit-rates. Since their metric does not account for losses, it is complementary to ETX.

One solution to high link loss ratios is to improve the apparent loss ratio with some form of redundancy. Forward error correction, MAC-level acknowledgment and retransmission, and solutions such as Snoop-TCP [3] and Tulip [24] all take this approach. Even with these techniques it is preferable to use low-loss-ratio rather than high-loss-ratio links: retransmissions (or other redundancy) reduce useful link capacity and generate interference.

A number of existing ad hoc wireless routing algorithms collect per-link signal strength information and apply a threshold to avoid links with high loss ratios [8, 9, 10, 11, 14, 17, 22, 28]. This approach may eliminate links that are necessary for connectivity, or fail to distinguish accurately between links; both of these are likely to be issues if many links have intermediate loss ratios.

Wireless Quality-of-Service (QoS) algorithms approach route selection from the top down. Some techniques explicitly schedule transmission slots in time or frequency division MAC layers to provide bandwidth guarantees [7, 13, 21, 23, 34], while others treat the MAC as opaque, and rely upon it for bandwidth and delay information and constraints [6, 30, 31]. These approaches are only successful if the lower layers can provide accurate information about the actual links, such as the average number of usable transmission slots, or the achievable throughput. However, none of these approaches consider the case of lossy links.

This paper assumes that the loss ratio of a given link cannot be controlled by the system. More sophisticated hardware might allow transmit power levels to be changed to make links better behaved. Existing systems exploit this idea, often with a focus on minimizing the energy consumption required to successfully deliver data [12, 18, 29].

7. Conclusions

This paper introduces a new metric for multi-hop wireless networks, called ETX. Route selection using ETX accounts for link loss ratios, the asymmetry of the loss ratios in the two directions of each link, and the reduction of throughput due to interference among the successive hops of a route. Measurements on a wireless test-bed show that ETX finds routes with significantly higher throughputs than a minimum hop-count metric, particularly for paths with two or more hops.

Several aspects of ETX could be improved in the future: its predictions of loss ratios for different packet sizes, particularly for 802.11b ACKs; its handling of networks with links that run at a variety of bit-rates; and the robustness of ETX probes when competing with high levels of data traffic.

The protocol implementations described in this paper are available at http://www.pdos.lcs.mit.edu/grid.

8. Acknowledgments

We are grateful to all the people in our laboratory who were willing to host wireless nodes. We thank Eddie Kohler for his help with, and improvements to, Click; Audun Tornquist for providing the initial DSR implementation; and Ben Chambers for his help with an earlier version of this work.

9. References

[1] The Network Simulator — ns-2, 2003. `http://www.isi.edu/nsnam/ns`.

[2] Baruch Awerbuch, David Holmer, and Herbert Rubens. High throughput route selection in multi-rate ad hoc wireless networks. Technical report, Johns Hopkins University, Computer Science Department, March 2003. Version 2.

[3] Hari Balakrishnan, Venkata N. Padmanabhan, Srinivasan Seshan, and Randy H. Katz. A comparison of mechanisms for improving TCP performance of wireless links. *IEEE/ACM Transactions on Networking*, 6(5), December 1997.

[4] Brahim Bensaou, Yu Wang, and Chi Chung Ko. Fair medium access in 802.11 based wireless ad-hoc networks. In *First Annual IEEE and ACM International Workshop on Mobile Ad Hoc Networking and Computing (MobiHoc)*, August 2000.

[5] Josh Broch, David A. Maltz, David B. Johnson, Yih-Chun Hu, and Jorjeta Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Proc. ACM/IEEE MobiCom*, pages 85–97, October 1998.

[6] Shigang Chen and Klara Nahrstedt. Distributed quality-of-service routing in ad hoc networks. *IEEE Journal on Selected Areas in Communcations*, 17(8), August 1999.

[7] T.-W. Chen, J.T. Tsai, and M. Gerla. QoS routing performance in multihop, multimedia, wireless networks. In *Proceedings of IEEE ICUPC '97*, 1997.

[8] Kwan-Wu Chin, John Judge, Aidan Williams, and Roger Kermode. Implementation experience with MANET routing protocols. *ACM SIGCOMM Computer Communications Review*, 32(5), November 2002.

[9] Brian H. Davies and T. R. Davies. The application of packet switching techniques to combat net radio. *Proceedings of the IEEE*, 75(1), January 1987.

[10] Rohit Dube, Cynthia D. Rais, Kuang-Yeh Wang, and Satish K. Tripathi. Signal stability-based adaptive routing (SSA) for ad hoc mobile networks. *IEEE Personal Communications*, February 1997.

[11] Tom Goff, Nael B. Abu-Ghazaleh, Dhananjay S. Phatak, and Ridvan Kahvecioglu. Preemptive routing in ad hoc networks. In *Proc. ACM/IEEE MobiCom*, July 2001.

[12] Wendi Rabiner Heinzelman, Anantha Chandrakasan, and Hari Balakrishnan. Energy-efficient Communication Protocols for Wireless Microsensor Networks. In *Proceedings of the Hawaiian International Conference on Systems Science*, January 2000.

[13] Yu-Ching Hsu, Tzu-Chieh Tsai, Ying-Dar Lin, and Mario Gerla. Bandwidth routing in multi-hop packet radio environment. In *Proceedings of the 3rd International Mobile Computing Workshop*, 1997.

[14] Yih-Chun Hu and David B. Johnson. Design and demonstration of live audio and video over multihop wireless ad hoc networks. In *Proceedings of the MILCOM 2002*.

[15] David B. Johnson. Routing in ad hoc networks of mobile hosts. In *Proc. of the IEEE Workshop on Mobile Computing Systems and Applications*, pages 158–163, December 1994.

[16] David B. Johnson, David A. Maltz, and Yih-Chun Hu. The Dynamic Source Routing protocol for mobile ad hoc networks (DSR). Internet draft (work in progress), IETF, April 2003. `http://www.ietf.org/internet-drafts/draft-ietf-manet-dsr-09.txt`.

[17] John Jubin and Janet D. Tornow. The DARPA packet radio network protocols. *Proceedings of the IEEE*, 75(1), January 1987.

[18] Eun-Sun Jung and Nitin Vaidya. A power control MAC protocol for ad hoc networks. In *Proc. ACM/IEEE MobiCom*, September 2002.

[19] Eddie Kohler, Robert Morris, Benjie Chen, John Jannotti, and M. Frans Kaashoek. The Click modular router. *ACM Transactions on Computer Systems*, 18(4), November 2000.

[20] Jinyang Li, Charles Blake, Douglas S. J. De Couto, Hu Imm Lee, and Robert Morris. Capacity of ad hoc wireless networks. In *Proceedings of the 7th ACM International Conference on Mobile Computing and Networking*, pages 61–69, Rome, Italy, July 2001.

[21] Chunhung Richard Lin. On-demand QoS routing in multihop mobile networks. In *Proc. IEEE Infocom*, April 2001.

[22] Henrik Lundgren, Erik Nordström, and Christian Tschudin. Coping with comunication gray zones in IEEE 802.11b based ad hoc networks. In *5th ACM international workshop on Wireless mobile multimedia (WoWMoM 2002)*, September 2002.

[23] Anastassios Michail and Anthony Ephremides. Algorithms for routing session traffic in wireless ad-hoc networks with energy and bandwidth limitations. In *Proceedings of 12th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, 2001.

[24] Christina Parsa and J. J. Garcia-Luna-Aceves. TULIP: A link-level protocol for improving TCP over wireless links. In *Proc. IEEE Wireless Communications and Networking Conference 1999 (WCNC 99)*, September 1999.

[25] Charles E. Perkins and Pravin Bhagwat. Highly dynamic Destination-Sequenced Distance-Vector routing (DSDV) for mobile computers. In *Proc. ACM SIGCOMM Conference (SIGCOMM '94)*, pages 234–244, August 1993.

[26] Charles E. Perkins and Elizabeth M. Royer. Ad hoc On-Demand Distance Vector Routing. In *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*, 1999.

[27] Rice Monarch Project. Wireless and mobility extensions to ns-2. `http://www.monarch.cs.rice.edu/cmu-ns.html`.

[28] Ratish J. Punnoose, Pavel V. Nitkin, Josh Broch, and Daniel D. Stancil. Optimizing wireless network protocols

using real-time predictive propagation modeling. In *Radio and Wireless Conference (RAWCON)*, August 1999.

[29] Ram Ramanathan and Regina Rosales-Hain. Topology control of multihop wireless networks using transmit power adjustment. In *Proc. IEEE Infocom*, March 2000.

[30] Samarth H. Shah and Klara Nahrstedt. Predictive location-based QoS routing in mobile ad hoc networks. In *Proceedings of IEEE International Conference on Communications*, 2002.

[31] Prasum Sinha, Raghupathy Sivakumar, and Vaduvur Bharghavan. CEDAR: A core-extraction distributed ad hoc routing algorithm. In *Proc. IEEE Infocom*, March 1999.

[32] Audun Tornquist. Modular and adaptive ad hoc routing in Click. Master's thesis, University of Colorado, 2001.

[33] Mark Yarvis, W. Steven Conner, Lakshman Krishnamurthy, Jasmeet Chhabra, Brent Elliott, and Alan Mainwaring. Real-world experiences with an interactive ad hoc sensor network. In *Proceedings of the International Workshop on Ad Hoc Networking*, August 2002.

[34] Chenxi Zhu and M. Scott Corson. QoS routing for mobile ad hoc networks. In *Proc. IEEE Infocom*, June 2001.