SrcRR: A High Throughput Routing Protocol for 802.11 Mesh Networks (DRAFT)

Daniel Aguayo, John Bicket, Robert Morris
{aguayo, jbicket, rtm}@lcs.mit.edu

ABSTRACT

This paper addresses a set of serious performance problems observed on the CityMesh 50-node 802.11b urban rooftop network that existing routing protocols do not consider. The original CityMesh routing protocol, based on DSR and ETX, delivered median TCP throughput of 20 kilobytes per second, despite the use of radios that could drive most individual links at much higher rates. The key problems included varying link loss rates, transient bursts of losses on otherwise high-quality links, poor transmit bit-rate selection at the 802.11 level, failure to identify high throughput routes in the multiple bit-rate network, and interference between bulk data traffic and routing protocol updates.

The paper describes SrcRR, a new routing protocol that includes solutions to these problems. SrcRR extends ETX by predicting the best 802.11 transmission bit-rate on each link. SrcRR monitors the loss rates of the links in each path it is using, as well as the loss rates of nearby alternate links, to ensure that it continues to use the best route. SrcRR switches to a new route only if the new route's metric is significantly better than that of the existing route. SrcRR performs more aggressive link-level retransmission than 802.11, and does not switch routes in response to modest numbers of lost packets. SrcRR uses its own transmit bit-rate selection algorithm based on medium-term loss rate measurements, replacing the algorithm built into the radio firmware.

The paper presents an experimental evaluation of SrcRR on CityMesh. SrcRR improves the median bulk TCP throughput between pairs of nodes from 20 KB/s to 110 KB/s. SrcRR achieves these improvements primarily by reducing the loss rate visible to TCP, which avoids TCP timeouts and consequent idle time, and by improved choice of link transmission bit-rate.

1. INTRODUCTION

This paper presents techniques to route and forward bulk data efficiently on multi-hop mesh wireless networks with stationary nodes. A typical use of such networks is to provide last-mile Internet access, either commercially [1, 17, 4] or in the form of community mesh networks sharing a few wired connections [18].

One such network, CityMesh, is a 50-node 802.11b rooftop network in an urban environment. The first-generation routing protocol on CityMesh was based on DSR [14] with the ETX routing metric [12]. Given the "static" nature of the network, one might expect that an ad hoc routing protocol such as DSR would be more than sufficient to provide good performance.

When the original CityMesh protocol was used for bulk TCP transfers, however, performance was disappointing. The median TCP throughput in the network was about 20 kilobytes per second, even though most links in the network could provide far higher bit-rates when used individually.

This paper pinpoints the problems which caused CityMesh to perform poorly. The problems include the routing metric's inability to incorporate link bit-rate predictions; poor bit-rate selection on individual links; transient bursts of packet loss on otherwise high-quality links, which result in link-level transmission failures; and collisions between data packets and routing broadcasts. None of these problems is specific to the protocols CityMesh used; indeed, any mesh network designer will probably face similar issues if high-throughput, bulk data transfer is a goal.

The paper presents techniques which address each of these issues. They are implemented and evaluated in the form of a new routing protocol, SrcRR. SrcRR uses DSR-like reactive queries to find routes initially, and a variant of ETX to predict which routes are likely to work the best.

SrcRR contributes the following new techniques:

- It uses an adaptive transmit rate control algorithm that performs much better than the one common in 802.11b cards, and chooses routes in cooperation with that algorithm.
- It detects failing routes with continuous measurement, rather than with 802.11 transmission failure, to avoid being misled by transient bursts of errors.
- It monitors the loss rate of alternate links for quick fail-over without the expense of a full flooded query.
- It uses heuristics to avoid switching routes due to interference between data and routing packets.

As a result of these techniques, SrcRR improves the median TCP throughput on CityMesh to about 110 kilobytes/second — a factor of five improvement.

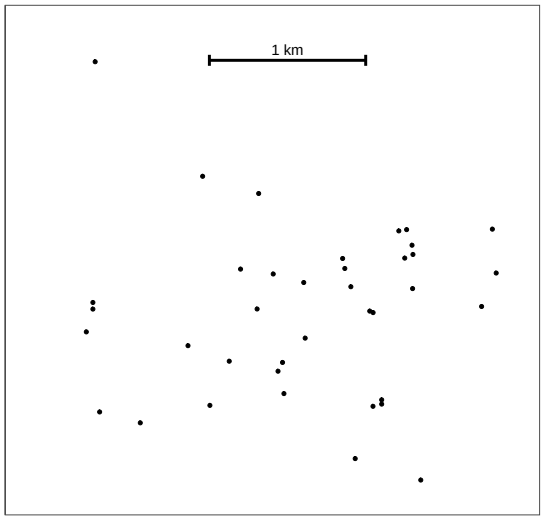

Figure 1: A scale map of CityMesh, with the black dots representing the nodes that participated in the experiments for this paper. [An underlying city map, showing city blocks, streets and buildings, is omitted for anonymity.]

The structure of the paper is as follows. Section 2 provides an overview of the CityMesh network. Section 3 discusses some of the specific performance problems found when using the first-generation protocol based on DSR with ETX. Section 4 presents the techniques that address each of these problems. Section 5 describes some implementation details. Section 6 evaluates the performance of SrcRR. Section 7 discusses related work, and Section 8 concludes.

2. CITYMESH OVERVIEW

SrcRR was developed for the CityMesh[1] network, which also serves as the experimental environment in which this paper evaluates SrcRR. CityMesh provides last-mile Internet access to graduate students living in an urban area around a university. In order to maximize the growth of the network, the equipment is designed so that each user can install his or her own node. In particular, nodes are equipped with omni-directional antennas which require no aiming. Users mount the antenna on the roof, run cable into their apartment, and plug in their node, a dedicated PC with an 802.11b card.

The network is contained in an urban, residential area of about four square kilometers, dominated by tightly-packed three and four story homes. Trees, other homes, schools, churches and office buildings commonly obstruct line-of-sight signal propagation between nodes. Most of the antennas are mounted on the chimneys of three- or four-story homes. Seven of the nodes are on taller buildings. The network includes three gateways to the wired Internet; most nodes must forward packets through multiple hops in order to reach the nearest gateway.

Each of the 50 CityMesh nodes uses an Engenius 802.11b card based on the Intersil Prism 2.5 chip-set. The cards are configured with RTS/CTS off, because experiments do not show that turning it on improves performance. The omni-directional antennas on the rooftops provide 8 dBi of gain with a 20-degree -3 dB vertical beam-width, and the intermediate cabling and lightning arrestors introduce an attenuation of 6 to 10 dB, depending on the length of cable. For all the measurements in this paper, the cards transmit using +23 dBm (200 milliwatts) transmission power.

[1]Name changed for anonymity.

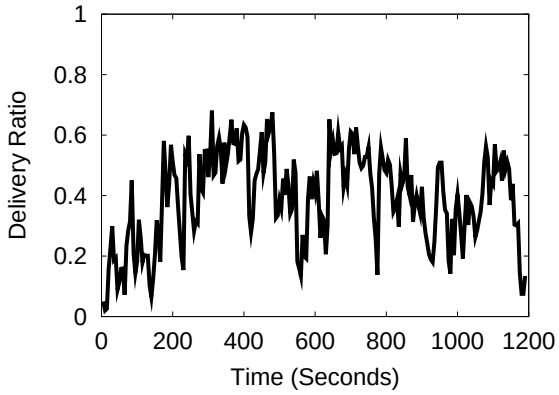

Figure 2: Delivery rate over time for a sample link. The y-axis indicates the fraction of packets delivered, averaged over 5-second intervals. The x-axis indicates time in seconds. Even without mobility, link loss rates change enough that routing protocols need to adapt to link quality changes.

3. PROBLEMS

This section presents the performance problems observed in early versions of CityMesh, along with the underlying causes of those problems.

CityMesh requires an adaptive routing protocol even though the nodes are stationary. Figure 2 shows how the fraction of packets delivered varies over time for a sample pair of nodes, measured by one of the nodes sending 1000-byte 802.11b broadcast packets as fast as possible at a bit-rate of eleven megabits per second for 1200 seconds. The values are averages over five-second intervals. The data show that the delivery ratio changes significantly over short periods of time, so the routing protocol must be able to switch routes on the same time scale.

Initially CityMesh used the DSR [14] routing protocol, modified to use the ETX metric [12]. DSR was chosen because almost all CityMesh nodes will only ever want to route data to the gateway nodes. As such, using a full link-state protocol would incur a great deal of unnecessary overhead.

ETX continuously measures each link's loss rate using periodic broadcast probes, and uses the loss rates of a route's links to predict how many total transmissions would be required to deliver a packet along the route (including link-layer retransmissions). This prediction is the ETX metric for the route. The route with the lowest ETX metric should in principle deliver the highest throughput. The rest of this paper will refer to this initial protocol as DSR+ETX, more details of which are presented in Section 4.

Using DSR+ETX, interactive performance of the CityMesh

network was good, with latency of about ten milliseconds per hop, and end-to-end losses (after 802.11 retransmissions) of only a few percent. However, bulk TCP transfers often had very low throughput, even over routes whose links individually delivered high throughput, and even accounting for the fact that only one node along a multi-hop path can typically transmit at a time. The end-to-end loss rate experienced by TCP during bulk transfers was often high, regardless of whether RTS/CTS was enabled. For example, one two-hop path delivered 40 kilobytes/second to TCP with DSR+ETX, with a 10% loss rate observed by TCP; testing each of the links in isolation suggested that the route should have been able to carry at least 120 kilobytes/second with essentially no losses.

Closer investigation revealed four underlying causes, described qualitatively in the next four sub-sections. Section 6 provides a quantitative assessment of their relative importances by evaluating the effect of solutions.

3.1 Transient Packet Loss Bursts

DSR+ETX frequently re-queried and switched routes due to link-level transmission failure, even though the links it was using had low average loss rate. A transmission failure means that the 802.11 radio sent a packet eight times without receiving an 802.11 link-level acknowledgment. In such a situation the node that just failed to forward the packet sends a DSR+ETX Route Error message to the sender, which deletes the offending link from its link-state cache and runs Dijkstra's algorithm to find a new route from that cache. If no such route exists, the sender floods a fresh query.

The intuition behind the Route Error mechanism in the original DSR is that repeated failures suggest that the next-hop node has moved out of radio range, or has crashed. Close observation of DSR+ETX on CityMesh showed that typically the radio link to the next hop was still working, but had merely suffered a transient burst of errors. Typically the alternate route that DSR+ETX switched to had lower throughput than the original route would have provided (this was verified in a number of examples by disabling Route Error messages). Of course sometimes persistent errors do indicate that propagation conditions have changed and that the link is no longer useful, so they cannot be completely ignored.

The source of the error bursts is not known. They may be caused by interference among the different hops in the route that DSR+ETX is using, or they might result from other 802.11 users on overlapping channels. Routing based on link quality measurements, such as those performed by ETX, does not necessarily aim to avoid links exhibiting these error bursts, because the best route choice may involve imperfect links.

3.2 ETX Metric Fragility

ETX measures link loss rate with periodic broadcast probes. When the network is idle, the probe loss rate is likely to be a good predictor of data loss rate. However, when the network is 100% busy with data packets, the data packets can cause a high fraction of the periodic probes to be lost due to collisions. This is not a serious problem for the data packets, which 802.11 re-sends, but 802.11 does not re-send the broadcast probes and ETX counts them as lost. As a result, ETX may assign low-quality metrics to links that actually deliver a high proportion of data packets. This may have no immediate effect on routes in active use, but may cause a subsequent DSR+ETX re-query to avoid what is actually the best route. Again, enabling RTS/CTS does not seem to solve this problem.

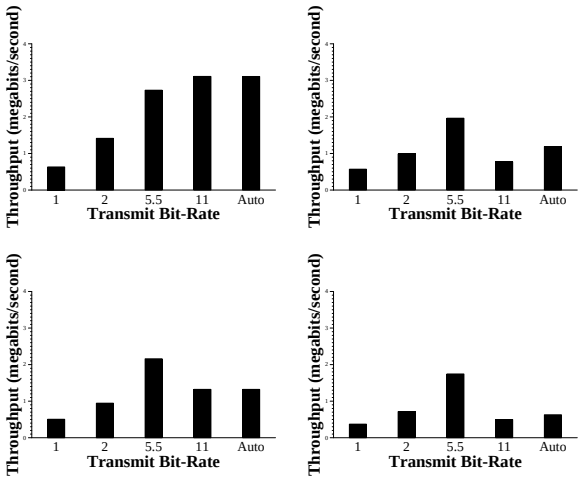

Figure 3: The relationship between transmission bit-rate and throughput for four different CityMesh links, using 1500-byte UDP packets. The throughputs are less than the bit-rates due to lost packets, 802.11 retransmissions, and 802.11 overhead. The Auto bar corresponds to the Prism 2.5 firmware's automatic rate selection algorithm; it often results in much less than the best possible throughput.

3.3 Bit Rate Selection

The Prism 2.5 802.11 firmware selects the transmission bit-rate in the following way [3]. The firmware maintains a current bit-rate for each station it talks to, which starts at 11 megabits/second. If a packet fails to be acknowledged after eight transmissions at the current bit-rate, the firmware reduces the current bit-rate to the next lowest. If a packet is ACKed, and no subsequent failures occur, and ten seconds elapse, the firmware returns to 11 megabits/second.

Figure 3 shows the throughputs that result from manual selection of each rate on four different CityMesh links, as well as the throughput resulting from using the Prism 2.5 firmware algorithm. The firmware algorithm works well on links that lose few packets at 11 megabits/second. On most other links, the firmware provides significantly less throughput than the ideal. One possibility is that the cards are sending more slowly than they should, because they are too sensitive to transient bursts of errors. It is also possible that the Prism algorithm sometimes runs links at too high a bit-rate; a link that drops 60% of packets at 11 megabits might have more throughput at 5.5 megabits, but might not drop the eight packets in a row required for the firmware to fall back.

The Prism algorithm may be justified in an environment in which losses are due only to low signal-to-noise ratio because of attenuation. In that case, most links would be either perfect or non-existent at any given bit rate, since the Prism manufacturer asserts that the range of S/N ratios in

which links are marginal is very narrow [2]. However, loss in outdoor 802.11 networks appears to be dominated by multi-path inter-symbol interference, not low S/N, which results in large numbers of marginal links [10, 11]. With marginal links it is not sufficient to choose the transmit bit-rate based on a simple working-vs-broken decision.

3.4 Bit-rate Aware Routing

ETX assigns metrics based on broadcast probes sent at 1 megabit/second, and does not explicitly favor links that would work well at higher bit-rates. ETX does have a slight bias in favor of potentially high bit-rate links, since it penalizes links that are lossy at 1 megabit and thus are unlikely to work well at higher bit-rates. However, it cannot distinguish among the links that are loss-free at 1 megabit, only some of which will work well at higher rates. As a result, routes did not take much advantage of high bit-rate links.

4. DESIGN

This section discusses the design of SrcRR. It starts by describing the basic protocol. It then introduces a sequence of improvements that address the problems described in the previous section. Each improvement has a shorthand name which will be used in the evaluation section.

4.1 Baseline SrcRR

The basic operation of SrcRR is similar to DSR with link caches: SrcRR is a reactive routing protocol with source-routed data traffic. [9]

Every node running SrcRR maintains a link cache, which tracks the ETX metric values for links it has heard about recently. Whenever a change is made to the link cache, the node locally runs Dijkstra's weighted shortest-path algorithm on this database to find the current, minimum-metric routes to all other nodes in the network. To ensure only fresh information is used for routing, if a link metric has not been updated within 30 seconds it is dropped from the link cache.

When a node wants to send data to a node to which it does not have a route, it floods a *route request*. When a node receives a route request, it appends its own node ID, as well as the current ETX metric from the node from which it received the request, and rebroadcasts it. A node will always forward a given route request the first time it receives it. If it receives the same route request again over a different route, it will forward it again if the accumulated route metric is better than the best metric it has forwarded so far. This ensures that the target of the route request will receive the best routes.

When a node receives a route request for which it is the target, it reverses the accumulated route and uses this as the source-route for a *route reply*. When the original source node receives this reply, it adds each of the links to its link cache, and then source-routes data over the minimum-metric path to the destination.

When a SrcRR node forwards a source-routed data packet, it updates its entry in the source route to contain the latest ETX metric for the link on which it received the packet. This allows the source and destination to maintain up-to-date link caches, and discover when a route's quality has declined enough that an alternate route would be better. In addition, each data packet includes a field to hold one extra link metric; a forwarding node will randomly, with probability $\frac{1}{n}$, where n is the number of nodes in the route, fill in that field with the ETX metric to one of its neighbors. This allows the source and destination to learn of the existence and metric of some alternate links. As with all changes to the link cache, this prompts recomputation of all the best routes using Dijkstra's algorithm.

All query and data packets contain ETX metrics for the links they have traversed so far. Any node that receives such a packet (including forwarding nodes) copies those metrics to its link cache.

Baseline SrcRR broadcasts a 300-byte ETX probe packet at randomized intervals averaging every ten seconds. ETX measures the loss rate from each neighbor by counting the fraction of probes received over the last three minutes (18 probes).

4.2 Ignoring Link Failure

The first improvement to SrcRR, hereafter referred to as IGNORE-FAIL, makes the protocol resistant to transient periods of high packet loss. If IGNORE-FAIL is disabled, SrcRR sends a route error message after the first transmission failure notification, causing the sender to switch to a different route. When IGNORE-FAIL is enabled, however, SrcRR does nothing special when the device indicates a 802.11 transmission failure (i.e. no 802.11 ACK after 8 attempts).

The effect of IGNORE-FAIL is to give more control to the ETX metric. The SrcRR sender will make its routing decisions based only on long-term ETX measurements, not as a result of temporary link conditions.

4.3 Transmit Bit-Rate Control

The next improvement, RATE-CTL, overrides the transmit bit-rate control scheme built into the 802.11 firmware. A SrcRR node determines the best bit-rate to each of its neighbors, and explicitly tells the card to send data at that bit-rate.

To determine the bit-rate, each node periodically sends broadcast probes at each possible bit-rate, as an extension of the ETX mechanism. Each node records what fraction of probes it receives from each of its neighbors at each bit-rate. SrcRR estimates the throughput at each bit-rate by multiplying the bit-rate by the probe delivery rate at the throughput. SrcRR then sends data packets at the bit-rate with the maximum predicted throughput.

In our implementation, for each bit-rate, a probe (1, 2, 5.5 and 11 megabits per second) is sent at randomized intervals that average every ten seconds.

4.4 Persistent Retries

The next improvement, PERSISTENCE, aims to reduce end-to-end loss in order to avoid TCP timeouts and consequent under-utilization. When the 802.11 device indicates a transmission failure, SrcRR places the packet at the head of the output queue to the device instead of discarding it. This means that the device will soon attempt to send the packet again. SrcRR only gives up on a packet after giving it to the device 40 times (for a total of 320 transmissions). [2] When SrcRR gives up it sends a Route Error message to the source

[2]The chosen value of 40 for the retransmit limit is a mistake, since it would cause delays longer than a typical TCP timeout. We do not, however, believe that this limit was ever reached. For a final version of this paper, we would redo the experiments with a much smaller value.

(regardless of whether the IGNORE-FAIL feature is enabled).

Because the device maintains a small internal queue, the subsequent attempts for a given packet are interleaved with other packets. This helps avoid head-of-line blocking when packets from different routes are interleaved in the output queue. On the other hand, it can re-order packets, the subject of Section 4.6.

4.5 Route Damping

Route damping (DAMPING) prevents flapping among routes which have similar metrics. Once SrcRR chooses a route, it will only switch to a new route after five seconds, or if it finds a new route with an ETX metric at least one full expected transmission lower than the existing route's metric. That is, the new route must consist of links with significantly lower loss rates.

The original reason for damping was to reduce the impact of ETX probe packets being lost due to collisions with data traffic (see Section 3.2). Section 6.4 describes additional unexpected benefits of damping.

4.6 Packet Reordering at Receiver

SrcRR re-orders packets (REORDER) in order to correct out of order delivery introduced by its persistent retries. Out-of-order packet arrival can cause TCP to reduce its window or time out.

Each SrcRR source maintains a sequence number for each route it is using, and tags each packet with that route's next sequence number. Each forwarding node inserts newly arrived packets into its outgoing queue in a way that keeps sequence numbers ordered for the corresponding route.

At the far end of a route, the SrcRR destination re-orders arriving packets before handing them off to the IP layer. In order to do this, the destination must retain out-of-order packets until all preceding packets on the same route have arrived. If an out-of-order packet sits in the re-order queue for more than 500 milliseconds, the receiver releases it anyway. 500 milliseconds is longer than most delays within the network, but shorter than the minimum TCP retransmission timeout.

In order to avoid this re-ordering delay when the real problem is packet discard due to queue overflow, each SrcRR forwarding node remembers when it has discarded a packet on a particular route due to queue overflow. It then sets a "congestion" bit in the next packet on that route, which causes the destination to immediately release all the packet in the re-order queue for that route.

Each SrcRR node also detects and suppresses duplicate packets which could result from SrcRR's aggressive retry scheme. Each node keeps track of the last 100 data packet sequence numbers for each source route, and drops any duplicate incoming packets.

4.7 Big Probe Packets

ETX measures loss rates with 300-byte probe packets, while TCP bulk transfers send 1500-byte packets. This means that ETX probably underestimates loss rates. Assuming the underestimate is consistent, it may make little difference when comparing two individual links or two routes with the same number of links. However, the actual loss rate values are important when comparing routes with different numbers of hops. For this reason the SrcRR BIG-PROBES feature sends probes that are 1500 bytes long instead of 300 bytes.

In certain cases, it would be possible to modify SrcRR to use data packets directly to obtain delivery ratio estimates over a given link. This technique does not remove the need for probe packets, however; the protocol needs to use probes to evaluate links that are not currently in use.

4.8 Estimated Transmission Time

In order to favor routes with higher bit-rate links, SrcRR evaluates routes with an "estimated transmission time" (ETT) metric instead of ETX. The goal of the ETT metric is to estimate the amount of time that each packet will keep the radio medium busy; minimizing this per-packet time should maximize throughput. SrcRR sums link ETT metrics to form a route ETT metric.

As described in Section 4.3, each SrcRR node periodically sends broadcast probes at all 802.11b bit-rates and predicts the best possible throughput to each neighbor given each bit-rate's loss rate. When ETT is enabled, SrcRR multiplies for each neighbor the estimate of the highest possible effective throughput by the delivery probability of ACKs in the reverse direction, and inverts:

$$\mathrm{ETT} = \frac{1}{P(ack) \times r_t}$$

$$\mathrm{r_t} = max(r_1, r_2, r_{5.5}, r_{11})$$

$P(ack)$ is the probability of delivery of an ACK based on probe losses in the reverse direction, and r_t is the estimated throughput of broadcast packets in the forward direction at bit-rate t megabits/second. ETT predicts the loss rate for 802.11 ACK packets by broadcasting small probes consisting of just 32 bytes of Ethernet payload.

In total, when using ETT, SrcRR sends an average of five probe packets every ten seconds: one small probe at one megabit, and one 1500-byte probe at each of 1, 2, 5.5, and 11 megabits. ETT sends each of the five probes at independent random intervals averaging ten seconds.

5. IMPLEMENTATION

SrcRR is implemented using the Click modular software router [15], running on Linux 2.4.20. The SrcRR implementation consists of about 5,000 lines of new C++ code, in addition to Click's existing code for IP processing other standard functions.

SrcRR does not follow the DSR standard closely. SrcRR defines its own headers rather than using DSR-style IP header options, and encapsulates IP packets. SrcRR does not run the interface in promiscuous mode, does not reply to Route Requests from cached information, does not salvage packets, does not remove all packets to a failed destination from the output queue, does not shorten routes, and does not piggy-back Route Errors on subsequent Route Requests.

Click is loaded as a Linux kernel module, which allows for fine-grained control of packet queuing: the SrcRR Click code hands packets directly to the device driver, and gets 802.11 transmission failures back from the driver. SrcRR uses the HostAP driver [16] in 802.11 "ad-hoc" mode, with the AP features turned off. The HostAP driver has been modified to report transmission failure and to allow transmit bit-rate to be specified per packet.

6. EVALUATION

This section evaluates the extent to which SrcRR improves throughput, as well as the effectiveness of each of its techniques.

6.1 Experimental Setup

The measurements described throughout this section were taken on a 31-node subset of the CityMesh network. Most of the throughput data reported are the median throughput among 14 randomly selected node pairs; the pairs are the same in all of the tests presented.[3]

The following procedure was used to measure throughput between each pair. The routing protocol was reset and allowed to run on an idle network for three minutes, to let the route metric (ETX or ETT) estimator collect link statistics. The sender next sent 10 ping packets to the receiver to establish an initial route. The sending node then started a TCP connection to the receiving node, and sent as much data as it could. The receiving node terminated the experiment 30 seconds after it first saw the TCP connection initiated, and recorded how much data it received.

Most of the following evaluation compares throughput between pairs of different SrcRR variants (that is, different sets of enabled features). For each node pair, evaluations of the two protocol variants were conducted one after the other with only two minutes intervening, to minimize the effects of any environmental changes.

While CityMesh ordinarily provides Internet access to its users, this traffic was disabled during these experiments. It can be expected that there was an unknown amount of interference from other nearby users of 802.11.

6.2 Overall Improvement

Figure 4 compares baseline SrcRR against SrcRR with all features enabled (full SrcRR). The graph shows TCP throughput between each of the 14 node pairs: the bottom line shows throughput provided by baseline SrcRR, and the top line shows throughput provided by full SrcRR. The node pairs are sorted by the higher of the two values. Full SrcRR out-performs baseline SrcRR by a large margin for almost all the node pairs.

The median full SrcRR performance has five times higher throughput than the median baseline performance. This improvement is more easily seen in the left and rightmost bars in Figure 6.

All the pairs with throughput of over 200 KB/s are one hop routes; this includes node pairs 11 through 13. Pair 13 involves a link that can run at 11 Mb/s, and both SrcRR variants used this rate; the reason baseline SrcRR does slightly better is probably due to a change in link conditions between runs. Node pairs 0 through 10 used routes with hop counts ranging from 2 to 6.

One of the nodes in pair 2 is connected to the rest of the network through a single low-quality link. That link dominates the throughput and the choice of the rest of the route does not make much difference. For this pairs, route selection improvements such as ETT and BIG PROBES have little effect, and the modest improvement come from the other features that prevent TCP from going into timeout so often.

[3]We tested only 14 node pairs due to time constraints; we would evaluate a larger, more representative sample for a final paper version.

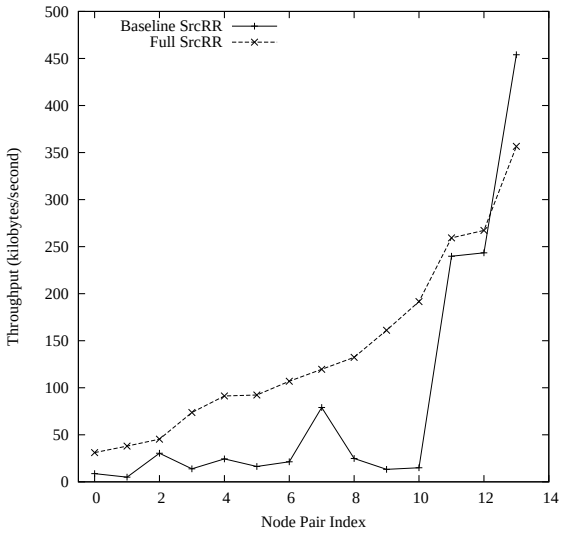

Figure 4: Bulk TCP throughput between each of the 14 CityMesh node pairs. The solid line represents baseline SrcRR, while the dashed line represents SrcRR with all features enabled (full SrcRR). Full SrcRR obtains the higher throughput for almost all of the pairs.

As an external point of comparison, Figure 5 compares the throughput of baseline SrcRR with the standards-compliant DSR in the Click distribution using the ETX metric. While not identical, the two routing protocols provide comparable throughputs.

6.3 Incremental Feature Addition

Figure 6 shows the TCP throughput provided by SrcRR as features are added one by one. The improvement provided by IGNORE-FAIL suggests that bursty losses are significant on high-quality links. The RATE-CTL result confirms that SrcRR does a better job of transmit bit-rate selection than the Prism firmware, and the ETT result shows the importance of bit-rate aware route selection.

Figure 8 sheds some light on why ETT is successful. It shows the distribution of metrics of links in the CityMesh network; each bar corresponds to one link, and the two ends indicate the ETT metrics in the two directions on that link. The asymmetry of many of the links validates the ETT design decision to measure loss rates with two different probe packet sizes to predict data and 802.11 ACK losses separately.

Figure 6 shows that PERSISTENCE has a negative impact by itself. The persistent retries, without REORDER, arrive at the TCP receiver out of order, trigger TCP's fast retransmission mechanism, and cause TCP to waste bandwidth with unnecessary re-transmissions.

BIG-PROBES alone is also detrimental. Increasing the ETX probe packet size causes ETX to predict the data packet loss rate more accurately, but the 802.11 ACK loss rate less accurately. BIG-PROBES is only useful when coupled with ETT's probing with both large and small sizes.

In total, the addition of the various improvements to SrcRR results in a factor of five improvement in the median through-

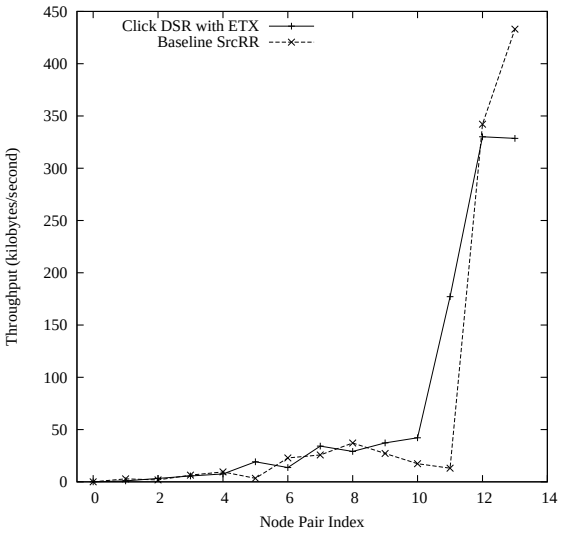

Figure 5: Throughput of baseline SrcRR compared with the Click standards-compliant DSR combined with ETX. The two protocols provide comparable throughputs.

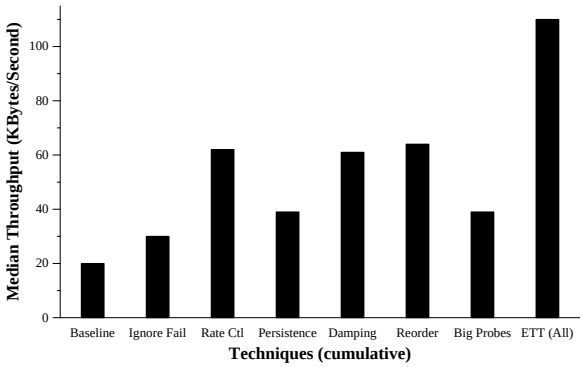

Figure 6: Cumulative effect on throughput of SrcRR features. Each bar shows the median throughput over the 14 node pairs; from left to right, each successive bar represents the cumulative addition of one more SrcRR feature. Rate control, damping, and ETT have the most effect. Persistent retries and large link probes have little effect by themselves, though Figure 7 shows that they are effective in combination with other features.

put over the 14 node pairs.

6.4 Feature Interdependence

SrcRR features PERSISTENCE and BIG PROBES decrease performance when they are added in Figure 6. To show that they are beneficial when combined with the rest of SrcRR, Figure 7 shows the effect on throughput of eliminating each individual feature from full SrcRR.

Eliminating PERSISTENCE decreases median throughput by 50%, because it eliminates losses that might otherwise cause TCP to time out; PERSISTENCE only works well when combined with REORDER. It's also the case that REORDER only works well when DAMPING is enabled, since the SrcRR re-order mechanism relies on the route not changing to allow it to distinguish out-of-order packets from packets lost due to queue overflow.

Turning off RATE-CTL decreases performance almost down to the level of Baseline SrcRR; the other features do little good if the links are running at dramatically reduced bit-rates. IGNORE-FAIL is almost as important, since without it SrcRR inappropriately switches away from the best routes. RATE-CTL works particularly well with IGNORE-FAIL and PERSISTENCE, since RATE-CTL intentionally operates links at high bit-rates that may cause significant losses.

7. RELATED WORK

SrcRR uses a variant of the ETX metric devised by De Couto et al. [12] SrcRR extends ETX to take link bit-rate into account, and to reflect the fact that 802.11 ACK packets are much smaller than data packets and thus are likely to have different probability of loss. SrcRR's ETT is similar to the *medium-time metric* proposed by Awerbuch et al [5]; one of ETT's advantages is that it takes losses into account.

The Receiver Based Auto Rate (RBAR) protocol [13] selects the transmit bit-rate for each 802.11 packet by observing the signal-to-noise ratio of the preceding RTS at the receiver. This approach works well if the nodes are mobile, so that longer-term measurements aren't appropriate, and if the relationship between S/N and error rate at each bit-rate is predictable. SrcRR is tailored for stationary nodes, so it can afford to make decisions based on recent loss rate history at the current transmit rate. More fundamentally, loss rates in the CityMesh environment do not appear to be predictable from S/N; this is consistent with the analysis by Clark *et al.* [10] showing that outdoor 802.11 error rates are dominated by multi-path inter-symbol interference, not S/N.

A number of solutions exist to improve the performance of TCP over lossy wireless networks [6, 8, 19], typically by hiding or repairing losses with TCP-specific mechanisms running in the wired gateway. The general lesson is that local repair is preferable to end-to-end TCP-level retransmission [7]. SrcRR adapts this idea to multi-hop networks with its aggressive link-level retransmission. It does not use TCP-specific knowledge, partially because that approach is not attractive when the wireless route can change during the lifetime of a TCP connection.

There are a number of other projects to build wireless mesh Internet access networks. For example, Wireless Leiden [18] is a 25-node 802.11 network spread over 25 square kilometers, using OSPF and the FreeBSD IP implementation for routing. The likely reason that the Leiden network can use routing techniques intended for wired networks is that their inter-node links are built with directional antennas. Much of SrcRR's complexity springs from CityMesh's use of omni-directional antennas and the resulting intermediate-quality links, but in principle it should be easier to expand CityMesh as a result.

8. CONCLUSIONS

This paper introduces SrcRR, a new routing protocol for multi-hop wireless networks. SrcRR finds high-throughput routes among multi-bit-rate links using an estimated transmission time routing metric, selects good bit-rates, and tol-

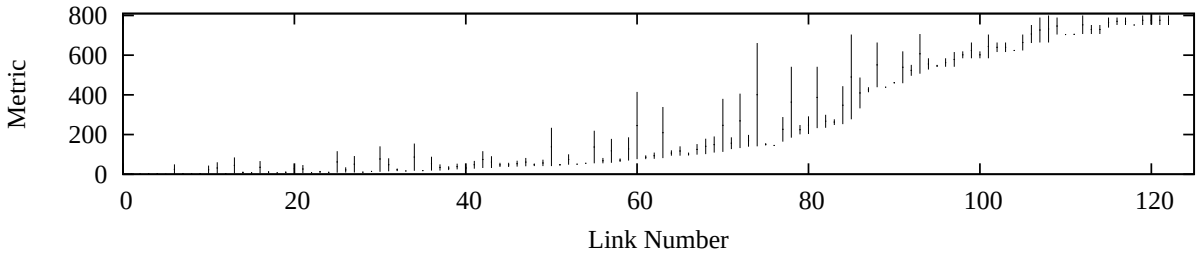

Figure 8: Link Metrics between each pair of hosts. The top and bottom ends of each vertical line indicate the metric in the two direction; the bars are sorted by the minimum of the two directions. The metric is determined by estimating the throughput for the link as described in Section 4.8. This graph shows that many links are asymmetric in quality and that there is a wide range of link quality. A metric of 100 represents a perfect 1 Mb/s bit-rate link, while 800 represents a perfect 11 Mb/s bit-rate link.

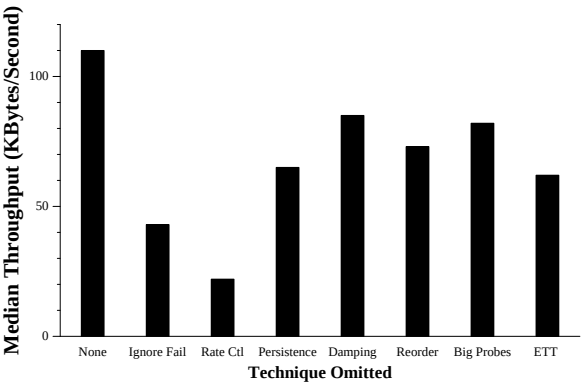

Figure 7: Full SrcRR compared to full SrcRR without each individual feature. The shorter the bar, the more the corresponding feature improves throughput when combined with all the other features. Using SrcRR's rate control and ignoring single link-level failures improves performance the most. Even persistent retries and large link probes, which do not help by themselves, do increase throughput when used in combination with the other features.

erates transient bursty packet loss. Measurements on the CityMesh metropolitan wireless mesh network demonstrate that SrcRR's technique increase bulk TCP throughput on multi-hop routes by a factor of five.

9. REFERENCES

[1] Nokia Rooftop wireless routing system. `http://www.nwr.nokia.com`.

[2] *HFA3863: Direct Sequence Spread Spectrum Baseband Processor with RAKE Receiver and Equalizer*. Intersil Corporation, 2000. Application Note FN4856.

[3] *Intersil PRISM Driver Programmer's Manual, Version 2.40*. Intersil Americas Inc., October 2002.

[4] Tropos networks technology whitepaper, May 2003. `http://www.troposnetworks.com/`.

[5] 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.

[6] A. Bakre and B. Badrinath. I-TCP: Indirect TCP for mobile hosts. In *ICDCS*, February 1995.

[7] H. Balakrishnan, V. Padmanabhan, S. Seshan, and R. Katz. A comparison of mechanisms for improving TCP performance over wireless links. In *Proc. ACM SIGCOMM Conference (SIGCOMM '96)*, August 1996.

[8] H. Balakrishnan, S. Seshan, and R. Katz. Improving reliable transport and handoff performance in cellular wireless networks. *ACM Wireless Networks*, 1(4), December 1995.

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

[10] M. V. Clark, K. K. Leung, B. McNair, and Z. Kostic. Outdoor IEEE 802.11 cellular networks: Radio link performance. In *Proc. of IEEE ICC 2002*, April 2002.

[11] Douglas S. J. De Couto, Daniel Aguayo, Benjamin A. Chambers, and Robert Morris. Effects of loss rate on ad hoc wireless routing. MIT-LCS-TR-836, MIT Laboratory for Computer Science, March 2002.

[12] Douglas S. J. De Couto, Daniel Aguayo, John Bicket, and Robert Morris. A high-throughput path metric for multi-hop wireless routing. In *Proceedings of the 9th ACM International Conference on Mobile Computing and Networking (MobiCom '03)*, San Diego, California, September 2003.

[13] G. Holland, N. Vaidya, and P. Bahl. A rate-adaptive MAC protocol for multi-hop wireless networks. In

Proc. ACM/IEEE MobiCom, July 2001.

[14] 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.

[15] 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.

[16] J. Malinen. Host AP driver for Intersil Prism. `http://hostap.epitest.fi/`.

[17] M. Ritter, R. Friday, R. Garces, W. San Filippo, and C-T. Nguyen. Mobile connectivity protocols and throughput measurements in the Ricochet microcellular data network (MCDN) system. In *Proc. ACM/IEEE MobiCom*, July 2001.

[18] Rudi van Drunen, Jasper Koolhaas, Huub Schuurmans, and Marten Vijn. Building a wireless community network in the Netherlands. In *USENIX/Freenix Conference*, June 2003.

[19] R. Yavatkar and N. Bhagwat. Improving end-to-end performance of TCP over mobile Internetworks. In *Mobile 94 Workshop on Mobile Computing Systems and Applications*, December 1994.