HxH: A Hop-by-Hop Transport Protocol for Multi-Hop Wireless Networks

Daniel Scofield
Computer Science Department
Brigham Young University
danscofield@byu.net

Lei Wang
Computer Science Department
Brigham Young University
lei@cs.byu.edu

Daniel Zappala
Computer Science Department
Brigham Young University
zappala@cs.byu.edu

ABSTRACT

TCP can perform poorly in multi-hop wireless networks due to problems that arise with contention and mobility. End-to-end protocols are at an inherent disadvantage in trying to solve these problems because their feedback loop operates over multiple wireless hops, which makes it difficult to diagnose problems that occur several hops away and hinders their ability to adapt to changing network conditions. In this paper, we design HxH, a hop-by-hop transport protocol that uses credit-based congestion control and reverse ACKs to solve many of the problems observed with TCP. We use a simulation study to demonstrate that our hop-by-hop approach allows HxH to respond quickly to changing network conditions and to exploit the unique characteristics of wireless networks to reduce overhead. We show that HxH greatly increases throughput and improves fairness, as compared to several end-to-end protocols, in both mesh and mobile ad hoc networks.

Categories and Subject Descriptors

C.2 [**Computer-Communication Networks**]: Network Architecture and Design—*Wireless communication*

1. INTRODUCTION

In this paper, we consider the problem of reliable transport across multi-hop wireless networks. We focus on networks that use a single radio in each node, using the IEEE 802.11 series of protocols, though we also discuss extensions for multi-radio networks. The network nodes may be primarily stationary, as with a mesh network, or mobile, as with an ad hoc network. We broadly classify both as multi-hop networks, since in each case connections must span several wireless hops, with the primary differences being that nodes may be mobile and may be power-constrained in the ad hoc case.

It is well known that TCP suffers from poor bandwidth utilization in multi-hop wireless networks. This problem arises from the many unique characteristics of wireless networks, including contention, interference, the hidden and exposed terminal problems, shared queues, half-duplex links, and route changes due to mobility. A wide range of research has attempted to solve this problem by providing feedback from intermediate nodes [8, 23, 10, 18], by modifying TCP's congestion control algorithm [6, 12, 1], or by designing new transport protocols [16, 2].

Our approach to solving this problem is to design a transport protocol that uses hop-by-hop rather than end-to-end protocol mechanisms. Our primary motivation is that hop-by-hop mechanisms free the transport protocol from having to use a feedback loop that spans multiple hops. Instead, each of the hops in a connection only has to worry about conditions that affect delivering packets to the next hop. This approach is particularly effective in wireless networks, because intermediate nodes are better positioned to react to contention and mobility-induced congestion, leading to a faster response and thus greater throughput. The main cost of the hop-by-hop approach is that it typically requires per-flow state in routers; this is less of a problem in wireless networks, since they typically operate at the edge of the network, with relatively few flows.

In designing a new transport protocol specifically for wireless networks, we have several goals. First, we want to use hop-by-hop mechanisms for both congestion control and reliability, while still retaining end-to-end transport. This enables us to explore how much benefit can be gained from a hop-by-hop design, so that we can evaluate whether the improved performance is worth the additional flow state. Second, we use 802.11 MAC mechanisms whenever possible, to eliminate inefficiencies due to duplicate functionality between the transport and link layers. Finally, the protocol should be general enough to improve throughput and fairness in both static and mobile wireless networks.

Our hop-by-hop transport protocol, called HxH, contains two components: a per-flow, credit-based congestion control algorithm, and end-to-end reliability using reverse ACKs. The advantage of using per-flow credits is that it eliminates congestion-induced loss, since a node will not transmit packets unless there are available credits for that flow at the next hop. Credit-based control also provides more stability than rate-based [5] and pricing-based [22] feedback, since flows are isolated from each other. With rate and pricing feedback, whenever new flows start the current rates or prices of all contending flows must be adjusted and then carried upstream. With credit-based feedback the only concern when adding a new flow is whether there is enough buffering avail-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
WICON'08 November 17-19, 2008, Maui, Hawaii, USA
Copyright 2008 ACM ICST 978-963-9799-36-3 ...$5.00.

able for the new flow; if per-flow buffering is kept small then this is not a problem on modern hardware.

While hop-by-hop congestion control has many benefits for wireless networks, pure hop-by-hop reliability is not as feasible. The fate-sharing principle states that it is acceptable to lose state for a conversation only if one of the entities in the conversation is itself lost [3]. If the wireless network was charged with delivering packets reliably, it would be very difficult to guarantee that packets would not be lost if nodes moved or crashed. Accordingly, we design a novel end-to-end reliability protocol that uses hop-by-hop, reverse ACKs to reduce ACK overhead. With reverse ACKs, feedback is carried upstream when data is sent downstream. This avoids the overhead of sending explicit transport-layer acknowledgments in most cases.

This paper makes several contributions relative to previous work in hop-by-hop congestion control for wireless networks. First, we design and evaluate credit-based congestion control, whereas existing work uses rate-based control, congestion pricing, or backpressure. Second, we design and evaluate the first reliable transmission protocol for wireless networks that uses hop-by-hop, reverse ACKs to provide end-to-end reliability. Existing work explores only congestion control and does not consider hop-by-hop approaches to reliable transmission. Third, we perform the first detailed simulation study to demonstrate the benefits of hop-by-hop transport as compared to end-to-end transport in both wireless mesh and mobile ad hoc networks.

Our simulation study compares the performance of HxH to TCP with Adaptive Pacing [6], which was designed for mesh networks; ATP [16], which was designed for ad hoc networks; and several variants of TCP. We show that HxH provides much greater throughput and better fairness than the end-to-end protocols when flows compete at a bottleneck. Because of its hop-by-hop congestion control algorithm, HxH is able to react to mobility much faster than an end-to-end protocol, and thus transfers more data when nodes move frequently. HxH also reacts quickly when congestion changes suddenly, whereas end-to-end protocols have difficulty converging to a new rate. We show that reverse ACKs can improve throughput by 70% for HxH as compared to explicit ACKs, and that this benefit holds even when passive feedback is not regularly received by nodes, due to contention or interference. We conclude by showing that these mechanisms combine to help HxH provide greater throughput in general mesh and ad hoc networks,

2. END-TO-END PROBLEMS

Many of the problems TCP encounters in wireless networks arise from its end-to-end design, which makes it difficult to diagnose problems that occur several hops away and to react quickly to changing network conditions. These problems include:

ACK Overhead: In a wireless network, the channel is a scarce resource that must be shared among all nodes within radio range. The transport protocol will operate most effectively if it can minimize the number of times the MAC layer must contend for access to this resource. Unfortunately, most end-to-end protocols rely on per-packet acknowledgments, which must contend for resources with data packets. As wireless networks become faster, channel access dominates the medium access time, and ACKs can consume as much as half of the wireless bandwidth.

There are several ways to reduce the impact of ACKs on transport performance. In TCP, ACKs can be piggybacked on data packets going in the reverse direction; however, many applications such as FTP, peer-to-peer, streaming video, and web traffic consist of primarily one-way flows. Delaying [4] or aggregating [16] ACKs can help, but these strategies run the risk of reducing responsiveness.

Improper Diagnosis of Loss: Wireless networks may lose packets for a number of reasons, including contention, interference, congestion, and mobility. End-to-end transport protocols may have difficulty distinguishing which type off loss has occured. If the transport protocol assumes all loss is due to congestion, then it will slow down needlessly when other types of loss occur.

This problem is sometimes solved with purely end-to-end mechanisms [18], but more frequently by using intermediate nodes to either hide the loss or to notify the transport protocol and put it into the correct state [8, 23, 10]. The node that has sent a frame unsuccessfully is clearly in the best position to determine what caused the loss.

Slow Feedback Loop: In order to prevent congestion, a transport protocol must react to changes in network conditions by adjusting its transmission rate. End-to-end protocols use some form of implicit or explicit signaling, such as packet loss or ECN. In wireless networks, there is a greater chance that this feedback loop can become delayed or lossy, due to MAC retransmissions, contention, collisions, and interference [16]. Irregular feedback results in the transport protocol making rate decisions with imperfect knowledge of network conditions.

Rate-based congestion control algorithms try to solve this problem by choosing a rate that minimizes congestion and contention-induced loss, thereby improving the regularity of feedback. Protocols such as TCP with Adaptive Pacing [6] provide higher goodput and better responsiveness than standard TCP congestion control algorithms. However, any end-to-end algorithm must still measure network conditions and receive feedback before reacting to congestion. The larger the delay in this feedback loop, the less responsive it will be to changes in the network.

Poor Response to Mobility: In wireless ad hoc networks, mobility may cause routes to fail and may even lead to network partitions. During these periods, TCP may time out if computing a new route takes too long. Even after the new route is found, TCP will begin with a small congestion window; if route changes happen frequently TCP may never send at a high rate.

Most work in this area ensures that TCP pauses during route changes, either by using feedback from intermediate nodes [10] or by inferring route changes from packet reordering events [18]. However, even with these changes, TCP may not react properly to mobility-induced congestion, which occurs when two flows whose paths did not previously cross suddenly are within interference or transmission range. Without additional help from intermediate nodes, end-to-end algorithms may have difficulty reacting to the sudden congestion that arises in this case.

3. HXH TRANSPORT PROTOCOL

The HxH transport protocol uses two main features to improve performance: credit-based congestion control at each hop and reverse acknowledgments to provide end-to-end reliability. We first discuss the HxH architecture, including

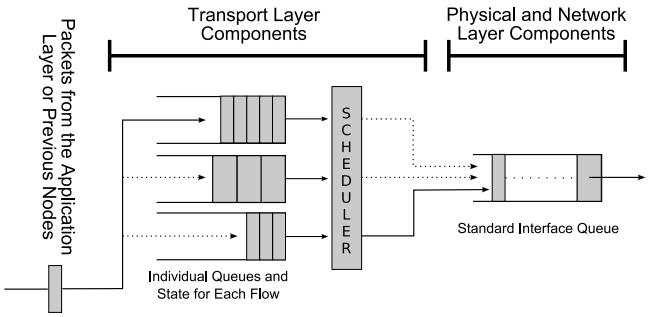


Figure 1: HxH Node Architecture

state maintenance and packet scheduling. We then describe how HxH implements congestion control and reliability using passive feedback whenever possible. This reduces overhead and allows HxH to operate over an unmodified MAC layer. We conclude by discussing MAC modifications that are necessary for multi-radio networks.

3.1 HxH Architecture

To provide per-flow congestion control and reliability, HxH uses the node architecture shown in Figure 1. Each node maintains state for each flow, consisting of a buffer, the amount of queue space available downstream (for congestion control) and a reverse ACK value (for reliability). A scheduler determines the order of packet transmission from each of the flow buffers, coordinating with the MAC layer to handle loss due to contention.

Flows are identified using the source and destination addresses and ports listed in the TCP and IP headers. When a node encounters a flow it has not seen before, it creates a new queue for the flow and initializes the flow's state to appropriate default values. HxH maintains the flow's queue and state as long as packets continue to be received for the flow. The state is deleted when it has not been used for a period of time. While collecting the results for this paper, we time out state after two seconds.

HxH uses per-flow fair queueing to provide fairness among flows that traverse the same path. This doesn't overcome all of the limitations of the 802.11 MAC; a protocol such as Neighborhood Red can be used to address these issues [21]. Whenever the scheduler transmits a packet, it checks with the MAC layer to ensure the transmission was successful. If the packet transmission fails, for example due to multihop contention [7], the scheduler places a copy of the packet back into the flow's queue and computes a new finishing time for the packet. This ensures that the scheduler won't repeatedly attempt to send packets from problematic flows without treating the other flows fairly.

HxH handles mobile hosts by suspending a flow when a route breaks. When a transmitting node detects the failure, it suspends flows that use the route until a new one is established. Nodes that were on the old path, but not the new one, continue to transmit so that the packets they hold are not lost. Nodes that were not on the old path, but are on the new one, establish state for the flow automatically.

3.2 Congestion Control

With credit-based congestion control, each node maintains a pool of credits for each flow, indicating the amount of buffer space available to that flow at the downstream node. Each time a packet is successfully transmitted for that flow, the pool is decremented. Each time the node overhears the next hop transmitting a packet, the pool is incremented. Transmitting from a node is allowed as long as the pool indicates there is space available downstream.

Because promiscuous listening is not completely reliable, the congestion control algorithm needs to periodically synchronize the number of available credits with its downstream neighbor. Credit-based algorithms for wired networks require each node to send a control message to its previous hop, telling it how many credits are available for the flow [13]. Wireless MAC protocols allow us to reduce the overhead of the credit feedback mechanism by instead using passive feedback to synchronize the credit pool. This also allows the controller to perform credit-based congestion control without modifying the MAC layer.

To synchronize the available credits, each HxH node includes the current number of credits for a flow in a shim header whenever it sends a packet for the flow. When the upstream node overhears the packet, it is able to synchronize the credit pool with this amount. As long as passive feedback succeeds a reasonable amount of time, this scheme works well. Our simulation quantifies the performance of HxH based on the success of passive feedback.

In extreme cases, it is possible that an upstream node will be unable to hear several consecutive packets sent downstream. When this occurs, it is possible for the upstream node to believe that there is no buffer space available when, in fact, the entire queue downstream is empty. Meanwhile, the node downstream is unable to correct its upstream neighbor, because it has no packets to transmit. To prevent flows from stopping in this case, upstream nodes will always transmit a packet from a stopped flow if the time since the last downstream transmission is more than five times the exponential mean weighted average of previous delays.

Because of this problem, it is also possible that a node may transmit a packet to a neighbor that has a full queue. To avoid this case, a downstream node under-reports its available buffer space by a few packets per flow. The extra buffer space is used to absorb errors when an upstream node transmits too soon.

3.3 Reliability

HxH eliminates most sources of end-to-end packet loss. It uses the 802.11 protocol to provide hop-by-hop reliability, and whenever a frame transmission fails it re-sends the frame again. When mobility changes the route for a flow, the mesh network nodes that were on the old route can still transmit their packets to the destination, as long as it remains connected to the mesh network. However, packet loss can still occur if mesh nodes crash or if a misbehaving timer allows the congestion control algorithm to repeatedly transmit to a downstream node even though it does not have sufficient buffer space.

Accordingly, HxH still provides end-to-end reliability, using reverse ACKs. HxH maintains a *reverse ACK* value for each of the flows passing through it. This value represents the *cumulative* sequence number that has been acknowledged by the destination. The last hop of a flow is in position to know whether the destination has successfully received a packet, since it gets a link-layer ACK. Thus each time the last hop successfully transmits a packet to the des-

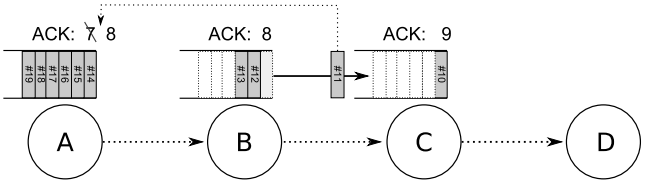


Figure 2: Reverse ACKS Using Passive Feedback

tination, it updates its reverse ACK value with the sequence number of this packet.

All other hops in the flow propagate the reverse ACK value upstream using passive feedback, as shown in Figure 2. Whenever a node sends a packet for a flow, it includes the reverse ACK value in the HxH shim header. When the previous node hears this value, it updates its own reverse ACK value.

As long as a steady stream of packets is moving along the flow, the ACK values will continue to be carried upstream. Eventually, the values reach the flow's source and the protocol responds just as if an explicit ACK was received; that is, it frees the acknowledged data from its send buffer.

3.3.1 Recovering From Loss

In the rare case that loss occurs, the last hop of a flow needs to take special care to be sure the reverse ACK value is updated properly. When packets arrive in order, the last hop can update its reverse ACK value each time it sends a packet, knowing that it reached the destination successfully. However, it cannot do this when there are gaps in the stream, as this voids the meaning of a cumulative ACK.

To solve this problem, if the last hop of a flow detects a gap, it stops updating the reverse ACK value and requests synchronization from the destination by setting a *sync* bit in the HxH shim header. When the destination sees the *sync* bit set, it determines the cumulative ACK value for the flow and sends this to the last hop, which synchronizes its state. The last hop will not continue updating the reverse ACK value until its synchronized state is equal to the highest sequence number it has transmitted.

To handle retransmission of lost packets, HxH uses end-to-end negative acknowledgments. If the destination detects a gap, it generates a negative ACK for that sequence number and sends it upstream to the source, which retransmits the packet. The destination also sets a timer, so that if it doesn't receive a retransmission in time it can resend the NACK.

3.3.2 Explicit Acknowledgments

Although this scheme is highly effective for FTP traffic or other bulk data transfers, additional methods are needed to handle bursty transmissions and the end of bulk transfers. Since the ACK data is carried upstream on data moving downstream, no ACKs will be sent if the flow stops transmitting. To handle these cases, HxH sets a bit in its shim header to specify that an explicit acknowledgment is requested. The source sets this bit on the last packet to leave the flow's buffer, requesting an explicit ACK from the destination.

Whenever the destination gets a packet with the explicit ACK bit set, it sends an explicit ACK upstream, subject to a rate limit. When forwarding explicit ACKs, intermediate nodes update their ACK state in the same way they would for a reverse ACK.

3.4 Multi-Radio Networks

In networks with multiple radios, an upstream node may not be able to use passive feedback, since the downstream node may transmit packets for the flow on a different frequency. In this case, we must modify the MAC layer so that the same information sent with passive feedback is instead conveyed to the previous node explicitly.

For the congestion control algorithm, the feedback a node requires is the number of credits available downstream. An upstream node must first tell the downstream node which flow the frame belongs to; it does this by including a flow identifier in the DATA frame. When the downstream node responds with an ACK frame, it includes the current number of credits for that flow.

For the reliability algorithm, the feedback a node requires is the reverse ACK value at the next hop node. The upstream node again includes a flow identifier in the DATA frame, and the downstream node includes the reverse ACK number for that flow in the ACK frame.

These modifications can be done fairly easily, since frequently a wireless MAC includes an ACK message already. Modifying the MAC has the added benefit that it simplifies the congestion control algorithm, since it eliminates the need for timers and extra buffer space that arise when using passive feedback. This likewise eliminates the need for a *sync* bit for the reliability algorithm, since the MAC layer modification provides explicit synchronization for each frame sent.

The performance of HxH when using a modified MAC will be equal or better to the performance when using passive feedback. When using passive feedback, performance will suffer whenever passive feedback becomes unreliable. The MAC modifications, on the other hand, ensure that explicit feedback is always delivered each time a frame is sent. As a result, our simulations evaluate a version of HxH that uses passive feedback exclusively. We then quantify the amount of performance degradation that occurs as passive feedback becomes unreliable.

4. RESULTS

We evaluate the HxH transport protocol using the *ns-2* simulator on a variety of topologies. Our primary comparisons are to TCP with Adaptive Pacing [6], an end-to-end transport protocol designed for mesh networks, and to ATP [16], a transport protocol designed for mobile ad hoc networks. In the case of ATP, an implementation was not available, so we wrote our own code based on the authors' paper. As a baseline, we also run all experiments with several variants of TCP. Surprisingly, Vegas and NewReno sometimes outperform TCP-AP and ATP. Vegas in particular has good performance due to its rate-based congestion control algorithm, which keeps smaller queues in routers and thus reduces some of the impact of multi-hop contention.

Many of our simulations use simple topologies. This is standard practice when evaluating transport protocols, as this allows researchers to focus on specific problems that affect performance, examine the dynamics of the congestion control algorithm, and make detailed comparisons between alternative protocols. We also include evaluations on more complex topologies, in order to assess overall performance when there are many competing flows and when nodes are mobile.

Unless otherwise stated, the simulations in this paper use

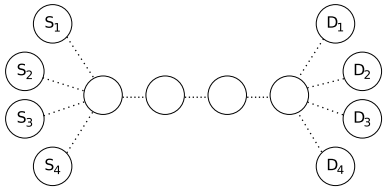


Figure 3: Dumbbell Topology

a wireless medium similar to the 802.11g specification. Link speeds are 54 Mbps, but the attainable throughput is much lower due to overhead incurred by the RTS/CTS exchange. The 802.11 MAC protocol specifies that the RTS and CTS control packets are sent at a speed of 1 Mbps, to provide backwards-compatibility among transmitters of differing maximum speeds. This lowers the capacity for a single hop to about 8.8 Mbps, with even lower speeds over multiple hops due to contention. Higher throughput is of course possible if the RTS/CTS exchange is also sent at 54 Mbps, but the important part of our methodology is that all tested protocols are run with the identical configuration.

Wherever topologies show adjacent nodes, they are placed so that they are within range of each other, but outside the range of non-adjacent nodes. We use the AODV routing protocol [14] for all simulations, even those without mobility, so that our methodology is consistent across all experiments. We use a packet size of 1000 bytes and plot throughput using a one-second sliding window, sampled every 100 milliseconds.

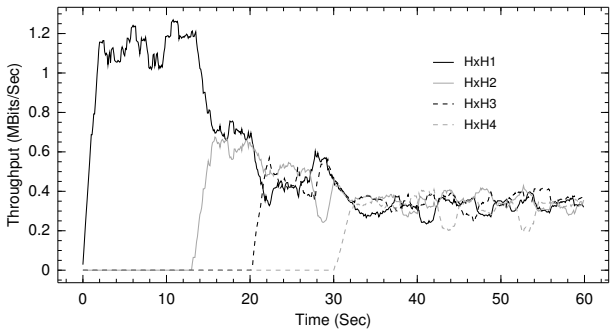
For HxH, nodes advertise a maximum queue size of 4 packets per flow. Because of this choice, in all simulations queue sizes are small and end-to-end delay is comparable to TCP Vegas, smaller than NewReno, and slightly larger than TCP with Adaptive Pacing.

4.1 Congestion and Fairness

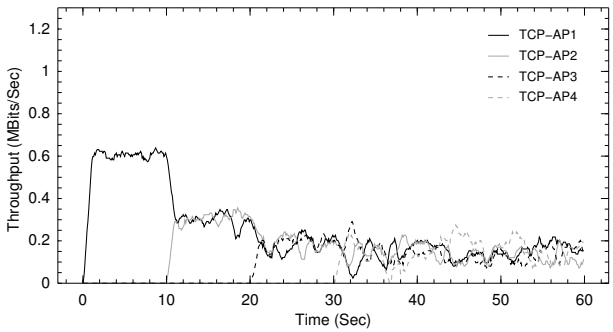
We first compare the performance of HxH with end-to-end protocols to examine their reaction to congestion and their ability to provide fairness. We use the dumbbell topology shown in Figure 3, with four flows sharing a four-hop link, causing a substantial bottleneck. We stagger the start time of each flow by ten seconds in order to observe the protocol's behavior when it must adjust its rate. We then plot the throughput observed by each of the four receivers.

Figure 4 highlights many of the benefits of using HxH. HxH achieves much higher throughput than the other end-to-end protocols due to its better handling of contention-induced loss and its use of reverse ACKs. When loss occurs due to contention, HxH simply re-queues the affected packet without reducing its rate. TCP with Adaptive Pacing is able to share bandwidth effectively, but has a lower overall throughput. ATP can occasionally achieve high throughput for one flow, due to aggregating ACKs into epochs, but reducing the amount of feedback also leads to instability.

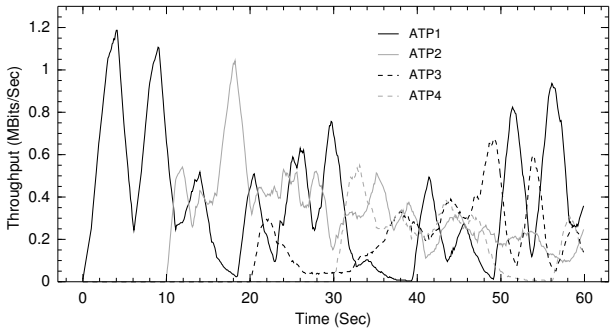
HxH also shares the bandwidth more fairly among competing flows. We use Jain's fairness equation to determine how fairly the available bandwidth was distributed among the four flows, in terms of bytes received. The results are summarized in Table 1, with the data shown representing only the last 30 seconds of the simulation, when all four flows are actively participating. TCP-AP is also able to allocate bandwidth fairly in this case, using only end-to-end



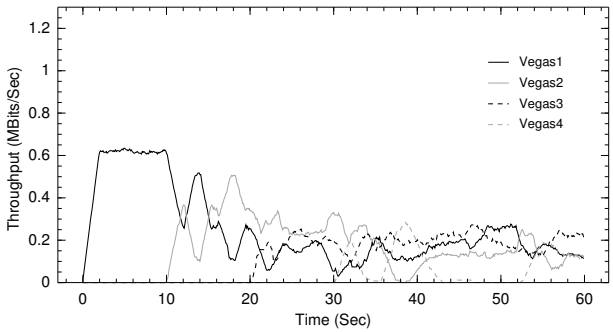
(a) HxH



(b) TCP-AP



(c) ATP



(d) TCP Vegas

Figure 4: Congestion Control on the Dumbbell Topology

	HxH	TCP-AP	ATP	Vegas	NewReno
Flow 1	1.21	0.55	1.04	0.61	0.38
Flow 2	1.33	0.52	0.93	0.50	0.72
Flow 3	1.31	0.51	0.99	0.73	0.59
Flow 4	1.20	0.57	0.82	0.32	0.57
Total	5.05	2.15	3.79	2.17	2.26
Fairness	0.998	0.998	0.993	0.928	0.956

Table 1: Fairness in the Dumbbell Topology

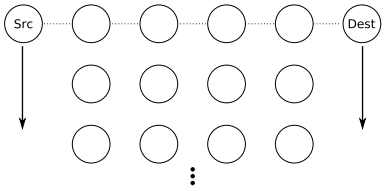


Figure 5: Mobile Lattice Topology

mechanisms, but its overall throughput is similar to other TCP variants.

Two small problems occur due to interactions with the 802.11 MAC. The throughput of HxH is initially a bit jagged; this occurs due to short-term unfairness in the 802.11 MAC [20]. In addition, with HxH (and Tahoe and NewReno), the second flow does not initially begin transmitting until several seconds after it should, due to the loss of an AODV route request message. A high rate of RTS/CTS traffic can prevent broadcast packets such as the AODV route request from being delivered [7, 12]. We plan to investigate solutions to these two problems in our future work.

4.2 Mobility

We next examine HxH and end-to-end protocols in a topology that stresses mobility-induced loss, without the effects of congestion. In the topology shown in Figure 5, the source and destination nodes periodically move along the edges of a lattice, inducing route failures at each step and requiring new routes to be formed. The transport protocols must cope with frequent route failure during this time. We repeat this simulation with different speeds for the source and destination, so that we can control the frequency of route failures the nodes experience. Each simulation is run for 60 seconds.

As shown in Figure 6, HxH is able to achieve a higher initial throughput and maintain the rate due to reverse ACKs and its hop-by-hop congestion control. For all protocols except ATP, throughput decreases as mobility increases. This is to be expected, since route failures require the protocols to wait while a new route is established. Yet, the TCP variants each suffer a drop of nearly 50%, while HxH only loses 8.2% of its total throughput. ATP performs better than the TCP variants in this scenario, but its instability, seen previously with the dumbbell topology, prevents it from doing even better.

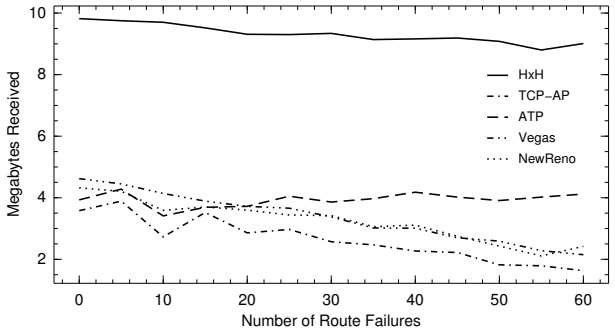


Figure 6: Performance on the Mobile Lattice Topology

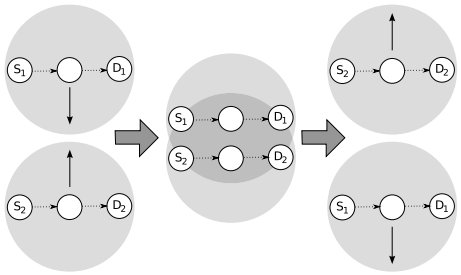


Figure 7: Topology with Sudden Congestion

4.3 Sudden Congestion

One of the differences between hop-by-hop and end-to-end congestion control is that the hop-by-hop approach can react more quickly to sudden congestion. To test this case, we use the topology shown in Figure 7. Two pairs of nodes communicate in this topology, initially positioned so that they are not within radio range and thus do not contend for bandwidth. As the simulation progresses, the two flows pass through each other's transmission ranges and eventually move back out of range of each other. This causes the flows to see rapid reductions and gains in their attainable throughput.

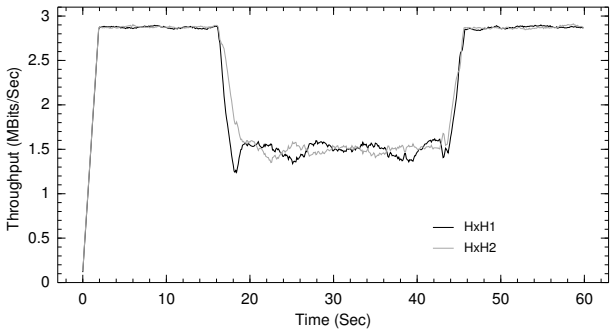
Figure 8 compares the results of HxH to those seen by TCP with Adaptive Pacing, which was the most effective end-to-end protocol in this scenario. HxH achieves nearly double the throughput of TCP-AP, primarily due to its use of reverse ACKs. In addition, hop-by-hop congestion control allows HxH to react to changing network conditions much more quickly, both when the flows meet and when they diverge.

4.4 Reverse Acknowledgments

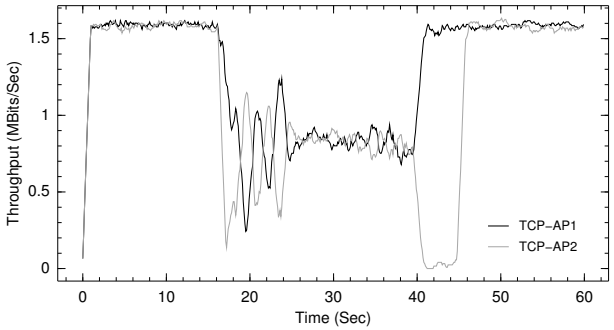
We use the same topology to determine the effect of reverse acknowledgments on HxH's performance. We create a version of HxH that uses explicit ACKs by disabling reverse ACKs and having the destination send an explicit end-to-end ACK upstream for every data packet it receives. This behavior mimics TCP's reliability mechanism, ensuring that any benefits seen are solely the result of HxH's congestion control algorithm.

The use of explicit ACKs does not hinder HxH's ability to react quickly to mobility – both the reverse and explicit ACK versions react similarly when the flows meet and then diverge. The primary difference is that using reverse ACKs allows for a throughput gain due to eliminating the channel accesses required by ACKs.

The benefit obtained by reverse acknowledgments varies with the speed of the wireless link. Because the RTS/CTS exchange occurs at 1 Mbps, to remain compatible with surrounding lower speed nodes, the overhead due to sending an explicit ACK for every packet increases up to about 50% as



(a) HxH



(b) TCP-AP

Figure 8: Reacting to Sudden Congestion

	1 Mbps	11 Mbps	54 Mbps
Reverse ACKs	0.80	3.89	5.63
Explicit ACKs	0.68	2.58	3.31
Gain	17.6%	50.8%	70.1%

Table 2: The Benefits of Reverse ACKs

the link speed increases. HxH takes good advantage of the extra bandwidth gained when eliminating explicit ACKs; as shown in Table 2, the gain varies from 17% to 70%.

4.5 The Effectiveness of Passive Feedback

The version of HxH we simulate relies heavily on the ability of nodes to passively overhear traffic sent by other nodes. It uses passive feedback to deliver the available credits and reverse ACK value from a downstream node to an upstream node. Accordingly, it is important to know how the ability to receive passive feedback affects its performance. Note that if HxH instead modifies the MAC to deliver explicit feedback, it is unaffected by this problem.

We conduct an experiment with a single flow on a five node chain, measuring the frequency with which a node in the chain is able to overhear the packets sent by its downstream neighbor. Without any intervention on our part, the success rate of passive feedback in this case is about 94%. We then emulate lower success rates by dropping passive feedback probabilistically, varying the success rate from 0 to 100%. Figure 9 plots HxH throughput as a function of the passive feedback success rate.

This result shows that HxH outperforms end-to-end protocols as long as passive feedback can be overheard at least 30% of the time. Performance does not drop significantly until the success rate drops below 80%. While these numbers may vary depending on the topology and workload, this is an encouraging result. For the other experiments presented in this paper, the success rate for passive feedback varies from 85% to 99%, without a significant impact on HxH performance.

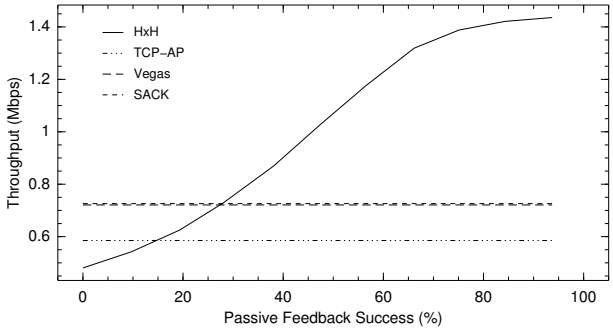


Figure 9: HxH Throughput vs Passive Feedback Success Rate

4.6 Mesh Network

To study performance in a mesh network, we use a six-by-six grid topology. We establish twelve flows, with six flows (1 - 6) sending from top to bottom, and six flows (7 - 12) from left to right. Since the simulation is meant to model a planned topology, no ad hoc routing protocol is used. Instead, each flow is given a route through the grid, which it maintains throughout the entire simulation.

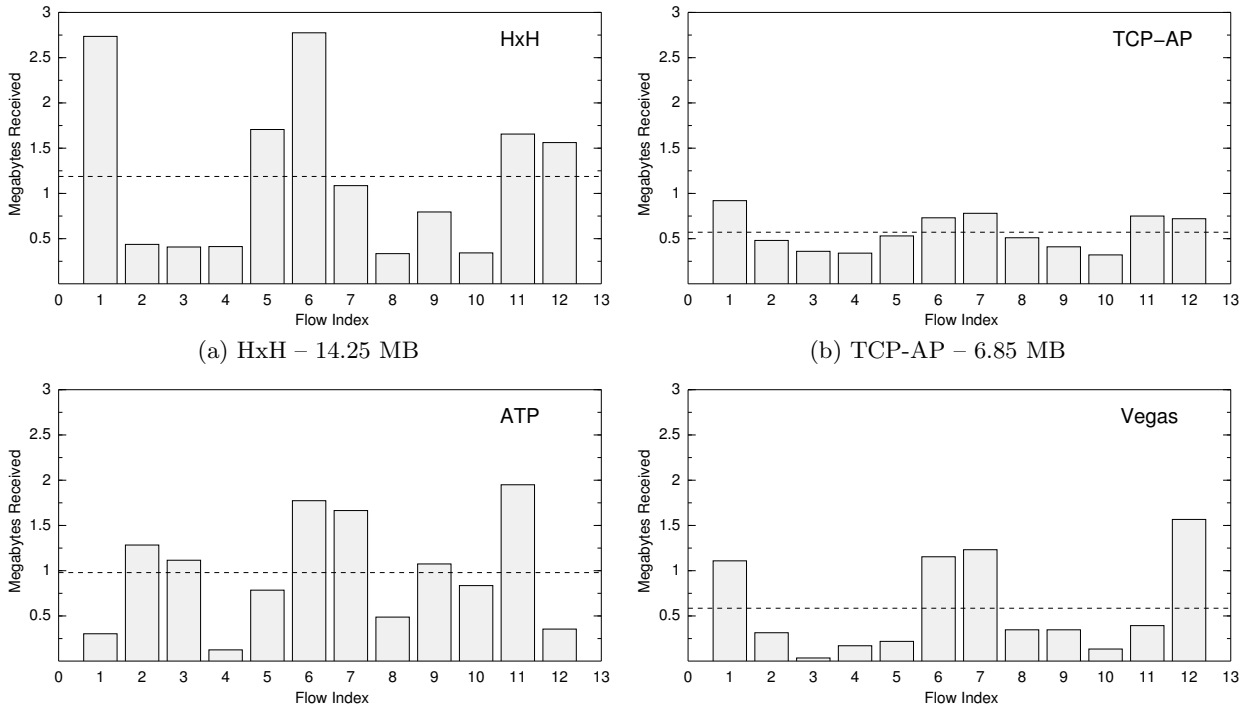
HxH and ATP both achieve higher throughput than the TCP variants due to the smaller number of explicit ACKs they send. Figure 10 shows the number of bytes received by each flow for HxH, TCP-AP, and ATP during a 60 second period. The other other TCP variants perform similarly to TCP-AP. The caption indicates the aggregate total bytes transferred by all flows during the simulation.

As can be expected, none of the protocols are able to achieve a completely fair allocation of bandwidth across all twelve flows, though TCP-AP does the best job. Although a transport protocol can worsen this problem, the unfairness seen is not exclusively a transport layer issue; the MAC layer and the topology are primarily responsible. Due to the shape of the topology, some flows compete with fewer flows than others. In addition, it is well-known that the 802.11 MAC is unfair in how it allocates bandwidth among competing nodes [21]. Although the transport layer cannot entirely fix the problem, the use of pacing appears to help in this situation.

4.7 Ad Hoc Networks

To model ad hoc networks, we generate 50 random topologies. We place 200 mobile wireless nodes in four square kilometers of terrain. The density of nodes within the terrain is intended to maintain a connected network. We create five flows among randomly-selected nodes, with connections lasting for 60 seconds. Meanwhile, all nodes move using a random-waypoint model with a speed of 10 meters/second and a pause time between 5 and 25 seconds.

As shown in Table 3, HxH again outperforms the other protocols, with HxH nearly doubling the throughput of its nearest competitor, TCP Vegas. This table shows the mean



(a) HxH – 14.25 MB

(b) TCP-AP – 6.85 MB

(c) ATP – 11.75 MB

(d) TCP Vegas – 7.02 MB

Figure 10: Performance in a Mesh Network

	HxH	TCP-AP	ATP	Vegas	NewReno
Mbps	2.676	1.025	1.069	1.488	1.423
MBytes	20.07	7.69	8.02	11.16	10.67

Table 3: Performance in Ad Hoc Networks

of the throughput and bytes transferred for each transport protocol, summed over all five flows. TCP-AP likely performs poorly in this case because it was designed for networks with only low mobility.

5. RELATED WORK

The earliest work on hop-by-hop congestion control was done for high-speed, wired networks. Mishra and Kanakia designed the HBH congestion control algorithm, in which a node measures the buffer occupancy for each connection and sends this information to the upstream node, which then regulates its sending rate to match the service rate of the downstream node [11]. Subsequent work for ATM networks devised credit-based flow control at each hop, with fair queueing to divide bandwidth among competing flows [13]. To protect against the possibility of losing credit messages, nodes re-synchronize periodically by measuring available capacity. We use similar concepts to these in our hop-by-hop architecture, with adaptations that capitalize on the unique aspects of wireless networks.

More recently, hop-by-hop congestion control algorithms have been designed for multi-hop wireless networks. In one approach, the network calculates congestion pricing, and a node regulates its sending rate based on the sum of the prices of all downstream nodes [22]. This work is only applicable to multi-radio networks and has only been evaluated with a static network and traffic flow; it is not clear how quickly the pricing can adapt to congestion that arises due to mobility and new traffic flows. Another approach limits the sending rate of each connection at a given node to one packet at a time [15]. The downside of this approach is that it is easy for the connection to underutilize available bandwidth. Any type of error, such as packet loss or inability to receive passive feedback, may result in the flow stopping to perform a retransmission.

In addition to this work, a number of hop-by-hop congestion control algorithms have been designed for wireless sensor networks [9, 17, 19, 5]. The hop-by-hop approach is very useful in sensor networks because intermediate nodes can react more quickly to transient congestion, and because algorithms must have simple, low-power implementations. While this work has been very successful, sensor networks have significant limitations, such as small buffers and packet sizes, low transmission rates, a many-to-one traffic pattern, and usually no mobility. Additional work will be needed to extend these algorithms to a more general multi-hop wireless network.

6. CONCLUSIONS AND FUTURE WORK

This work demonstrates the viability of credit-based, hop-by-hop transport for multi-hop wireless networks. Credit-based congestion control reacts quickly whenever network conditions change, and can improve fairness among flows competing on the same path. Reverse ACKs eliminate much of the overhead of end-to-end transport, while still providing end-to-end reliability. The primary advantage of end-to-end protocols is that they require less per-flow state, but in wireless networks the number of flows is typically small

enough to justify the performance improvement gained by providing per-flow, hop-by-hop congestion control.

In future work, we plan to evaluate hop-by-hop congestion control experimentally using a mesh network testbed, explore the proper settings for timers and buffer sizes, and study performance for short-lived flows. We will also expand our evaluation to include other hop-by-hop congestion control algorithms, including those that use rate-based and pricing-based feedback. In addition, though our architecture does provide fairness among flows that traverse the same path, we are interested in finding complementary techniques that provide fairness among competing nodes over larger areas. Finally, we are working on integrating the hop-by-hop approach with end-to-end protocols, so that it operates transparently over multiple wireless hops, while also interoperating with the wired Internet.

7. REFERENCES

[1] O. Akan and I. Akyildiz. ATL: An Adaptive Transport Layer Suite for Next-Generation Wireless Internet. *IEEE Journal on Selected Areas in Communications*, 22, June 2004.

[2] K. Chen, K. Nahrstedt, and N. Vaidya. The Utility of Explicit Rate-Based Flow Control in Mobile Ad Hoc Networks. In *IEEE WCNC*, 2004.

[3] D. Clark. The design philosophy of the darpa internet protocols. *SIGCOMM Comput. Commun. Rev.*, 18(4):106–114, 1988.

[4] R. de Oliveira and T. Braun. A Dynamic Adaptive Acknowledgment Strategy for TCP over Multihop Wireless Networks. In *IEEE INFOCOM*, 2005.

[5] C.-T. Ee and R. Bajcsy. Congestion Control and Fairness for Many-to-One Routing in Sensor Networks. In *ACM SenSys*, 2004.

[6] S. M. ElRakabawy, A. Klemm, and C. Lindemann. TCP with Adaptive Pacing for Multihop Wireless Networks. In *ACM MobiHoc*, 2005.

[7] Z. Fu, P. Zerfos, H. Luo, S. Lu, L. Zhang, and M. Gerla. The Impact of Multihop Wireless Channel on TCP Throughput and Loss. In *IEEE INFOCOM*, 2003.

[8] G. Holland and N. H. Vaidya. Analysis of TCP Performance over Mobile Ad Hoc Networks. In *Mobile Computing and Networking*, 1999.

[9] B. Hull, K. Jamieson, and H. Balakrishnan. Mitigating Congestion in Wireless Sensor Networks. In *ACM SenSys*, Baltimore, MD, November 2004.

[10] J. Liu and S. Singh. ATCP: TCP for Mobile Ad Hoc Networks. *IEEE Journals on Selected Areas in Communications*, 19(7):1300–1315, 7 2001.

[11] P. P. Mishra and H. Kanakia. A Hop by Hop Rate-Based Congestion Control Scheme. In *ACM SIGCOMM*, 1992.

[12] K. Nahm, A. Helmy, and C.-C. J. Kuo. TCP over Multihop 802.11 Networks: Issues and Performance Enhancement. In *ACM MobiHoc*, 2005.

[13] C. Özveren, R. Simcoe, and G. Varghese. Reliable and Efficient Hop-by-Hop Flow Control. In *ACM SIGCOMM*, 1994.

[14] C. Perkins and E. Royer. Ad Hoc On Demand Distance Vector (AODV) Algorithm. In *IEEE WMCSA*, 1999.

[15] B. Scheuermann, C. Lochert, and M. Mauve. Implicit hop-by-hop congestion control in wireless multihop networks. *Ad Hoc Netw.*, 6(2):260–286, 2008.

[16] K. Sundaresan, V. Anantharaman, H.-Y. Hsieh, and R. Sivakumar. ATP: a Reliable Transport Protocol for Ad-Hoc Networks. In *ACM MobiHoc*, 2003.

[17] C.-Y. Wan, S. B. Eisenman, and A. T. Campbell. CODA: congestion detection and avoidance in sensor networks. In *ACM SenSys*, 2003.

[18] F. Wang and Y. Zhang. Improving TCP Performance over Mobile Ad-Hoc Networks with Out-of-Order Detection and Response. In *ACM MobiHoc*, 2002.

[19] A. Woo and D. Culler. A Transmission Control Scheme for Media Access in Sensor Networks. In *ACM Mobicom*, 2004.

[20] K. Xu, M. Gerla, and S. Bae. Effectiveness of RTS/CTS Handshake in IEEE 802.11 Based Ad Hoc Networks. *Ad Hoc Networks*, 1(1):107–123, 2003.

[21] K. Xu, M. Gerla, L. Qi, and Y. Shu. Enhancing TCP Fairness in Ad Hoc Wireless Networks Using Neighborhood RED. In *ACM MobiCom*, 2003.

[22] Y. Yi and S. Shakkottai. Hop-by-Hop Congestion Control over a Wireless Multi-Hop Network. *IEEE/ACM Transactions on Networking*, 15(1), 2007.

[23] X. Yu. Improving TCP Performance over Mobile Ad Hoc Networks by Exploiting Cross-layer Information Awareness. In *ACM MobiCom*, 2004.