2007-04-17

Hop-by-Hop Transport Control for Multi-Hop Wireless Networks

Daniel N. Scofield
Brigham Young University - Provo

HOP-BY-HOP TRANSPORT CONTROL FOR MULTI-HOP WIRELESS NETWORKS

by

Dan Scofield

A thesis submitted to the faculty of

Brigham Young University

in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computer Science

Brigham Young University

April 2007

BRIGHAM YOUNG UNIVERSITY

GRADUATE COMMITTEE APPROVAL

of a thesis submitted by

Dan Scofield

This thesis has been read by each member of the following graduate committee and by majority vote has been found to be satisfactory.

Date	Daniel Zappala, Chair
Date	Mark Clement
Date	Kevin Seppi

BRIGHAM YOUNG UNIVERSITY

As chair of the candidate's graduate committee, I have read the thesis of Dan Scofield in its final form and have found that (1) its format, citations, and bibliographical style are consistent and acceptable and fulfill university and department style requirements; (2) its illustrative materials including figures, tables, and charts are in place; and (3) the final manuscript is satisfactory to the graduate committee and is ready for submission to the university library.

Date	Daniel Zappala Chair, Graduate Committee
Accepted for the Department	Parris K. Egbert Graduate Coordinator
Accepted for the College	Thomas W. Sederberg Associate Dean, College of Physical and Mathematical Sciences

ABSTRACT

HOP-BY-HOP TRANSPORT CONTROL FOR MULTI-HOP WIRELESS NETWORKS

Dan Scofield

Department of Computer Science

Master of Science

TCP can perform poorly in multi-hop wireless networks due to problems with contention and poor feedback from end-to-end control algorithms. This thesis explores the design of a hop-by-hop transport protocol (HxH). By allowing intermediate nodes to actively participate, the protocol can respond more quickly to changing network conditions and exploit the unique characteristics of wireless networks. Results indicate that hop-by-hop transport can achieve throughput rates that are double those of TCP, depending on the speed of the wireless links.

ACKNOWLEDGMENTS

Words alone could never adequately convey the many thanks I owe to those who have supported my academic efforts. The love and patience offered to me has been invaluable in ensuring not only the completion of this thesis, but also my happiness during the long years leading to it.

To Dianna, my dear wife, I offer my deepest appreciation. Your smiles and encouragement have meant the world to me. To my father, I extend my most heartfelt gratitude. You have done much more for me over the years than I think you will ever know. May this thesis be a tribute to your own keen mind and example.

To the rest, thank you all.

Contents

Chapter 1

Introduction

The *One Laptop per Child* (OLPC) project proposed by MIT hopes to place laptops in the hands of children around the world[1]. The project will help children to further their education by giving them constant opportunities to interact with modern technology. Cooperation from sponsors such as AMD, Marvell Semiconductor, Google, Nortel Networks, Red Hat, and others has allowed for the creation of specialized hardware and software that can meet the specific needs of the laptop while still being cost effective.

One challenge being faced by OLPC and its sponsors is providing effective network communications between the laptops. Many of the laptops are targeted for underdeveloped parts of the world, where no networking infrastructure exists. In such areas, it is impossible to connect to the Internet, but wireless technology can be used to communicate between computers in the same area.

These ad hoc networks differ from traditional wireless networks, where wireless technology is used across a single hop to connect to the Internet via an access point. As shown in Figure 1.1, distant laptops will communicate by relying on intermediate nodes to deliver their data. Individual nodes will sacrifice bandwidth to forward data for their peers, but will receive the promise that others will do the same for them.

[1]More detailed information on the OLPC project can be found on their website: http://www.laptop.org.

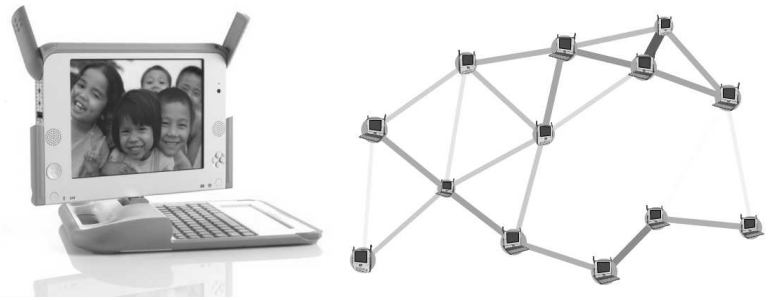

Figure 1.1: *Multi-hop Wireless Networks:* Through wireless technology OLPC laptops will be able to form networks dynamically and communicate.

Not all networks formed by the OLPC laptops will be ad hoc networks. While serving as governor, Mitt Romney requested funds to distribute the laptops to children in Massachusetts. Likewise, the laptops have been requested by governments in Brazil and other Internet capable countries. In such cases, wireless technology can be used to extend the reach of existing wireless access points. Allowing users to connect to access points that are multiple hops away reduces the cost required to provide network connections to large geographic areas, since the amount of cable that needs to be laid is minimized.

Although multi-hop wireless networks are useful, data transport in such networks is complicated by the current limitations of wireless communication. In particular, the Internet's transmission control protocol (TCP) suffers from two main performance problems in such networks:

1. Throughput is not fairly allocated among wireless nodes.
2. The use of explicit acknowledgment (ACK) packets incurs an unreasonably high overhead.

Thesis Statement: This thesis focuses on the development of HxH, a hop-by-hop transport protocol designed specifically for multi-hop wireless networks. HxH is able to overcome the challenges faced by TCP. It achieves a more fair allocation of bandwidth and higher throughput than TCP when running in the same topologies.

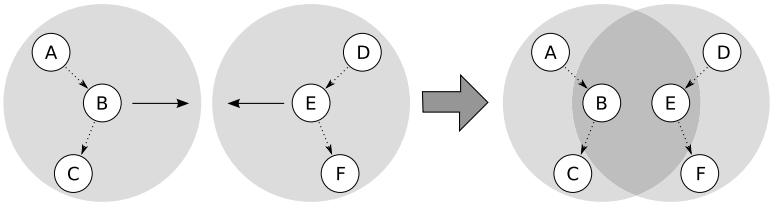

Figure 1.2: *Wireless Bandwidth Fluctuates:* Nodes in the same neighborhood must share bandwidth. As independent flows move into range of each other they contend for the same channel, immediately cutting their attainable bandwidths in half.

To do so, it uses novel approaches to flow control and reliability that allow transport control to occur without the need to explicitly send packets upstream.

The remainder of this chapter provides specific details about the problems seen by TCP. Chapter 2 presents existing research related to work done in this thesis. The HxH protocol is presented in Chapter 3. Chapter 4 details the results of simulations comparing HxH against existing protocols. Finally, Chapter 5 draws conclusions about the work in this thesis and outlines future work that would be beneficial to the research community.

1.1 Fairness Among Wireless Nodes

The first challenge faced by transport control protocols in multi-hop wireless networks is fairly distributing bandwidth among flows. This problem is complicated by node mobility and the current limitations of wireless communications. Unlike wired networks, all nodes within the same transmission range must share the wireless channel.

Since only one node may transmit at a time, the bandwidth available to individual nodes within a network has the potential to fluctuate dramatically with little or no warning. As nodes move within the network, their transmission ranges shift as well, affecting different sets of nodes. As shown in Figure 1.2, the bandwidth available to each node fluctuates as nodes move into and out of each other's transmission ranges.

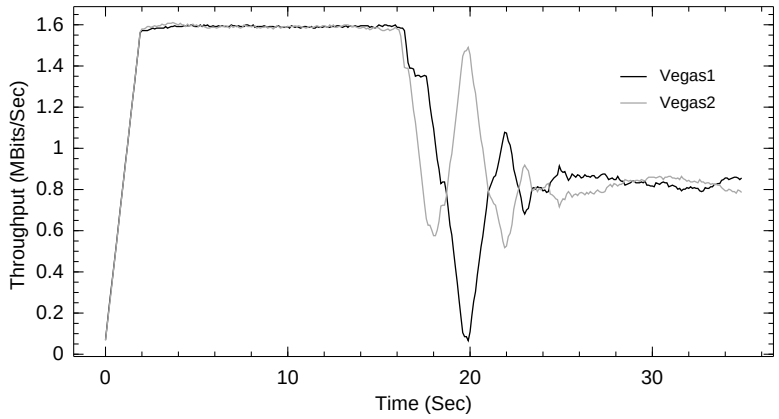

Figure 1.3: *TCP in a Mobile Topology:* TCP flows oscillate dramatically when independent flows suddenly come into each other's transmission ranges.

Figure 1.3 presents the results of running TCP in the environment shown in Figure 1.2. As the two flows come together, the bandwidth available to each is immediately cut in half. Unaware of the change, TCP continues transmitting at the same speed as before. Quickly, packets begin to be lost, causing TCP to reduce its transmission rate. The flows then attempt to converge to a fair allocation of bandwidth, but oscillate around the fair distribution for several seconds.

In addition to node mobility, conditions within multi-hop wireless networks are further complicated by additional ways in which packets may be lost. In wired networks, protocols have been designed with the assumption that all loss occurs due to congestion [10]. In wireless networks, packet loss can be caused not only by congestion, but also by node mobility and wireless interference [15].

These problems are intensified by the end-to-end nature of TCP. As shown in Figure 1.4, only the source and destination nodes are able to control the data transfer. Problems that occur at intermediate nodes cannot be addressed until their effects propagate to the destination node and feedback is sent to the source. Likewise, the state of the entire network must be inferred from information available at the ends.

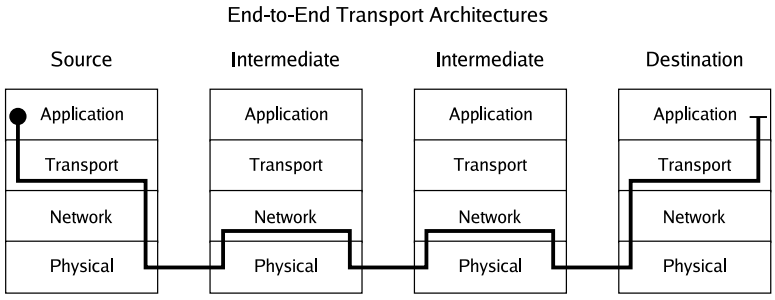

Figure 1.4: *End-to-end Design:* End-to-end architectures allow packets to pass through the transport layer at the source and destination nodes only. Problems that occur at interior nodes cannot be handled until they propagate to the source and destination nodes, introducing latency.

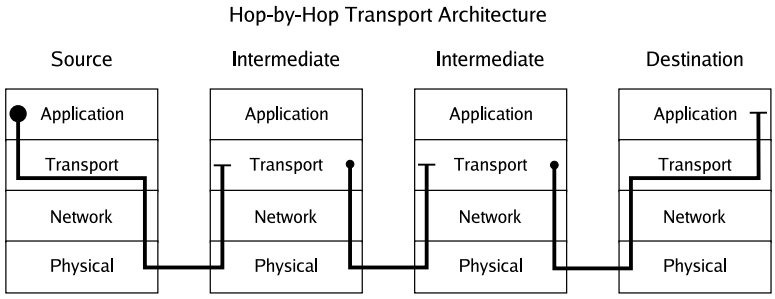

Figure 1.5: *Hop-by-hop Design:* Hop-by-hop protocols pass packets to the transport layer at every node. Rate adjustments can be made anywhere, allowing intermediate nodes to respond to adverse network conditions where they occur.

HxH addresses this issue by using a hop-by-hop flow control scheme. As shown in Figure 1.5, packets are delivered to the transport layer at each hop along the path. Unlike end-to-end protocols, which must infer network conditions based on incomplete information, each node in a hop-by-hop protocol has a complete view of all network conditions relevant to its transmissions. Nodes can observe all data transmission occurring in their wireless neighborhood, giving them a clear indication of how much bandwidth is available and how it must be shared between existing flows.

Even when mobility causes network conditions to change rapidly, nodes using HxH are able to detect the changes and react before packet loss occurs. Doing so

allows HxH to maintain throughputs for flows that are more fair than those produced by TCP. The specifics of the HxH flow control scheme are covered in Section 3.3.

1.2 Impact of Explicit Acknowledgment Packets

The second challenge faced by TCP in multi-hop wireless networks is an unusually high overhead incurred by acknowledgment (ACK) packets. To provide reliability, TCP transmits an explicit acknowledgment packet upstream for every data packet successfully received downstream. In wired networks, the overhead produced by ACK packets is negligible, since their size is dwarfed by the size of data packets. In multi-hop wireless networks, the ACK packets incur much larger overheads due to the use of channel reservations.

Channel reservations are an essential part of the IEEE 802.11 media access control (MAC) protocol used by nodes in wireless networks. Nodes wishing to transmit first send a *request to send* (RTS) message. Receivers act on these requests in a first come, first serve order. In response, they transmit a *clear to send* (CTS) message. The CTS informs the requesting node that the channel has been reserved, but also informs other nodes in the receiver's transmission range that they should wait to transmit until the packet has been delivered. The sender then transmits the data and the receiver sends a link layer acknowledgment to indicate that the packet was received without interference. Figure 1.6 outlines this process.

It is important to note that the link-layer acknowledgment sent by the 802.11 protocol is different than the ACK packets sent by TCP. TCP's ACK packets are sent end-to-end, from the destination node to the source, allowing the source to be sure that the packet has been received and providing round-trip timing information used by TCP. The link level ACKs provided by 802.11 function across a single hop only.

Reserving the wireless channel helps to minimize interference, but requires time. Equation 1.1 provides the calculation for determining how long it will take to

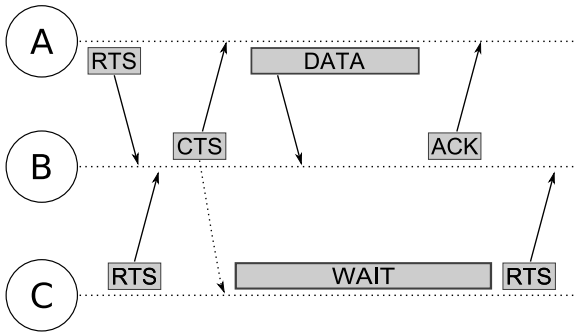

Figure 1.6: *Channel Reservation:* Nodes reserve the channel using a sequence of RTS/CTS/Data/ACK.

send a packet of arbitrary size ($pSize$) while using a wireless link of a given speed ($lSpeed$). Any transmission using channel reservations incurs a delay of $792\mu s$. This delay is the result of transmitting the RTS, CTS and ACK packets.

$$time = 792\mu s + \frac{pSize}{lSpeed} \tag{1.1}$$

The overhead incurred with each transmission varies with the size of the packet and the speed of the wireless link. Overhead increases as link speeds increase or packet sizes decrease. When using TCP, significant overhead is incurred by ACK packets because they are small (40 bytes) compared to data packets.

Equation 1.2 shows the proportional amount of the channel required to send end-to-end acknowledgment packets. This equation is plotted in Figure 1.7 with an ACK size of 40 bytes and data sizes of 1024, 2048 and 4096 bytes. Even at low speeds, the overhead required to send end-to-end acknowledgments is significant (4 - 13%). At high speeds, the amount of time needed to send acknowledgment packets nearly equals the amount of time available to send data.

$$overhead = \frac{792\mu s + \frac{ackSize}{lSpeed}}{2 \cdot 792\mu s + \frac{ackSize+dataSize}{lSpeed}} \tag{1.2}$$

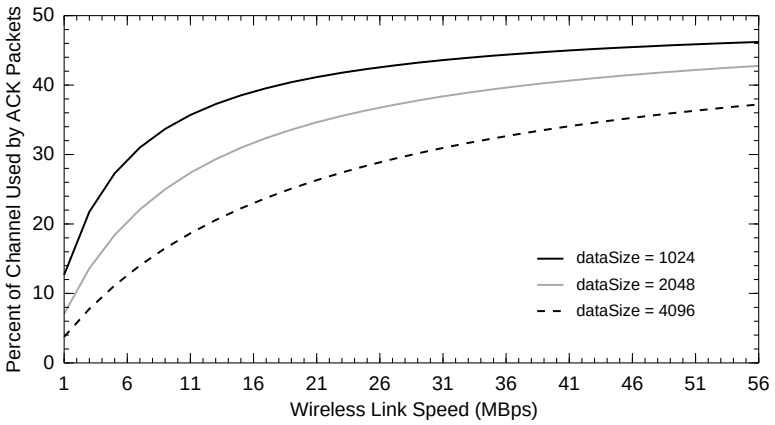

Figure 1.7: *Overhead for End-to-End Acknowledgments:* As link speeds increase, ACK packets begin to dominate the network. When using fast wireless links, the overhead is unacceptably high.

The reason the overhead required by ACK packets is able to increase so dramatically is that regardless of the speed of the link, the RTS/CTS delay is fixed. This occurs because the 802.11 MAC protocol specifies that these control packets will *always* be sent at a speed of 1 Mbps. Doing so provides the backwards-compatability necessary for areas with transmitters of differing maximum speeds [8].

The backwards-compatability provided by the MAC protocol is essential, since the environment in which nodes will communicate is unknown. It is possible that the entire network will be comprised of high speed nodes, especially in cases where ad hoc networks are formed among nodes with homogeneous hardware. Yet, even with the OLPC project, the possibility of laptops being used within the United States, Brazil, and other modern countries mandates cooperation with older hardware.

To overcome the significant overhead incurred by explicit acknowledgment packets, HxH sends ACKs passively. The protocol exploits the unique benefits of the wireless channel to carry acknowledgment information upstream on data moving downstream. The details of this technique are outlined in Section 3.4.1.

Chapter 2

Related Work

Performance problems with TCP in multi-hop wireless networks are well known in the research community. Issues of effective flow control and reducing transmission overhead have been the subject of many ongoing research projects.

2.1 Hop-by-hop Protocols

Recognizing the need to respond more quickly to the volatile nature of multi-hop networks, Yi and Shakkottai recently presented an argument for the viability of hop-by-hop protocols in multi-hop wireless networks [29]. Through proof and simulation, they demonstrated that a hop-by-hop protocol is not only possible in multi-hop wireless networks, but is also highly effective. Their protocol does not, however, address the drawbacks of sending explicit control messages upstream. HxH extends their ideas by focusing on developing a hop-by-hop algorithm that achieves effective flow control without the need for control packets.

Hop-by-hop transport protocols are not a new idea. In ATM networks, credit based hop-by-hop congestion control schemes have been studied by Özveren, et al. [20] and Kung, et al. [12]. Likewise, Mishra and Kanakia explored the use of hop-by-hop protocols in high speed packet switching networks [18]. In all cases, each node periodically transmits the status of its queues and packet servicing rates to its immediate neighbors. Neighbors use the information to control the rate at which

packets are sent downstream. By carefully controlling the transmission rate at each hop, loss is avoided and throughput is significantly increased. The primary drawback noted by all papers is the requirement to store and process state about each flow at intermediate nodes. Although HxH suffers the same drawbacks, the overhead is more manageable. Wireless networks typically have much fewer flows and transmit at much slower speeds than seen in the Internet, making hop-by-hop protocols more feasible.

2.2 ATP

In the domain of end-to-end protocols, researchers have also focused on the issue of reducing ACK overhead. The ad hoc transport protocol (ATP) presented by Sundaresan, et al. is one recent example [24]. ATP is designed to completely replace TCP in ad hoc networks. To reduce the overhead incurred by explicit ACKs, ATP keeps all acknowledgment data buffered at the receiver and sends a single cumulative ACK packet once per epoch (one second). ATP also differs from TCP in the following ways: First, ATP abandons slow start. Second, ATP detects congestion without inducing loss. Finally, ATP uses rate-based rather than window-based transmissions.

Although ATP outperforms TCP, it is unable to maintain stable transmission rates. Since ATP restricts rate-based decisions to once per epoch, network changes that occur during the epoch cause queuing to occur. Eventually, ATP detects the increasing delay and lowers its transmission rate. Frequently, ATP overcompensates and chooses rates that are much lower than necessary, resulting in periodic oscillations between high and low transmission speeds, as shown in Figure 2.1.

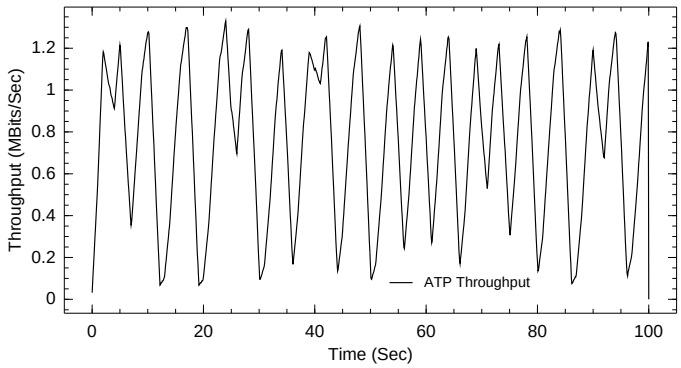

Figure 2.1: *ATP Instability:* ATP is shown running alone across a chain of five wireless nodes. ATP's epoch-based reliability reduces ACK overhead, but causes its throughput to oscillate severely.

2.3 Improving TCP Performance

In addition to developing new protocols, many researchers have focused on trying to directly improve TCP in wireless networks. Early attempts met with limited success. Most researchers initially focused on improving TCP's performance over the last hop between a wired network and a wireless node [1, 17]. Such techniques have been able to improve performance in networks with a single wireless hop, but are not applicable to multi-hop wireless networks.

In multi-hop networks, Holland and Vaidya noted that TCP's performance decreases significantly when mobility causes route failures [9]. Route failures occur when intermediate nodes along a communication path are no longer reachable, as shown in Figure 2.2. When this occurs, the source must send out a new route request before continuing to transmit. Packets are delayed while a new route is established. In severe cases, TCP's round-trip timers fire while waiting for a new route, causing TCP to reduce the rate at which packets are sent.

To reduce the performance losses incurred by route failures, Holland and Vaidya propose the use of early link failure notifications (ELFN). When routing failures occur, TCP-ELFN freezes its round-trip timers until a new route has been

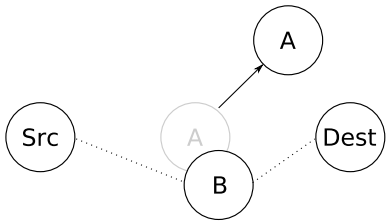

Figure 2.2: *Route Failures:* Mobility in multi-hop networks can cause routes to fail. In this case, when node A moves, the source and destination must find a new route through node B before continuing to transmit.

established. Although ELFN improves the performance of TCP, it only addresses performance losses when routing failures occur. Additional research has made similar incremental improvements to TCP [5, 7, 14, 19, 26]. As with ELFN, these solutions make only small improvements to isolated parts of TCP.

Unlike other protocols, best effort acknowledgment delivery (BEAD) attempts to improve TCP performance by modifying the network layer [30]. BEAD detects lost acknowledgments and retransmits them from the nearest node that successfully handled the data. Unfortunately, the increased load required to salvage lost packets is detrimental in some cases. BEAD responds to heavily congested networks by increasing traffic as it attempts to recover lost data. The increased load further deteriorates an already undesirable network state.

2.4 TCP-Vegas

Along with proposing enhancements for TCP, researchers have also noted that some versions of TCP perform significantly better than others. TCP-Vegas, presented by Brakmo, et al. [2], has been shown to outperform other traditional versions of TCP in multi-hop wireless networks [6]. Like ATP, Vegas improves performance by using rate-based transmissions responding to congestion before loss occurs. Unfortunately, like other versions of TCP, Vegas relies on a steady stream of acknowledgments, which incurs overhead.

Chapter 3

The HxH Protocol

3.1 Overview

To overcome the limitations seen by TCP in multi-hop wireless networks, HxH focuses on two main components: hop-by-hop flow control and end-to-end reliability. Flow control ensures that packets are only sent when there are enough resources available downstream to handle the packet. Likewise, it responds to rapidly changing network conditions and ensures that packets are sent downstream in a fair manner. End-to-end reliability allows the source and destination to be sure that the entire message has delivered successfully.

HxH takes a novel approach to both flow control and reliability. By utilizing the unique benefits of wireless networks, both schemes can be implemented without sending packets upstream. Instead, the information necessary to implement the schemes is overheard upstream from packets moving downstream.

To implement HxH's flow control and reliability schemes, a modified protocol stack for wireless nodes is needed. Section 3.2 describes a scheduling layer kept at each node to monitor all flows passing through it. The scheduler collects all information needed to correctly manage the flows and coordinates the efforts of the flow control and reliability schemes, which are described in Sections 3.3 and 3.4, respectively.

As with any network protocol, some information must be carried along with each packet in order to coordinate efforts between nodes. Figure 3.1 presents the

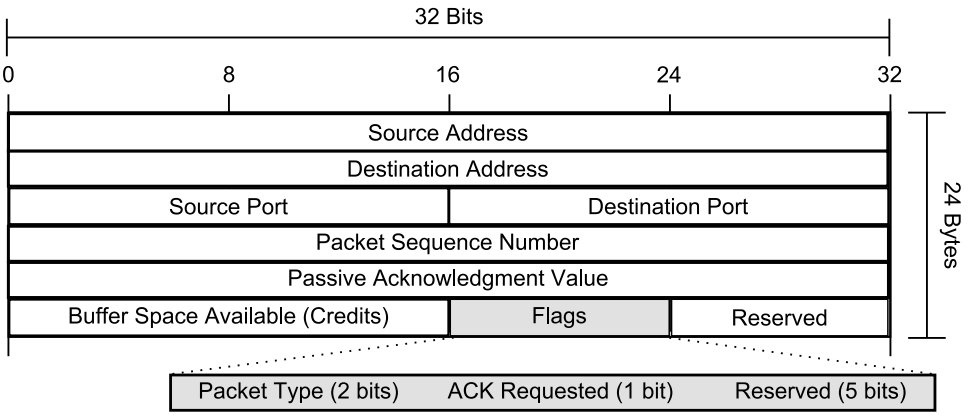

Figure 3.1: *HxH Packet Header:* The HxH packet header contains data necessary for efficient flow control and reliable packet delivery.

HxH packet header, which contains the required information. The particular values held within the header will be discussed in the sections that follow. In networks that are exclusively wireless, this is the only transport layer header needed. In hybrid networks, where multiple wireless hops are used to eventually connect to a wired network, this header would act as a shim, carried above a TCP header. The HxH header would be used over the wireless portion of the network, then would be removed at an access point before the packet was sent through the wired portion of the network.

3.2 The Scheduling Layer

To manage state and determine which actions to take for each flow, the HxH protocol adds a new layer to the protocol stack. The scheduling layer lies immediately beneath the transport layer and is responsible for collecting and handling the packets for all HxH flows passing through a node. Likewise, it maintains state about each flow, tracking the highest sequence number seen, the amount of queue space available downstream and the flow's current acknowledgment values.

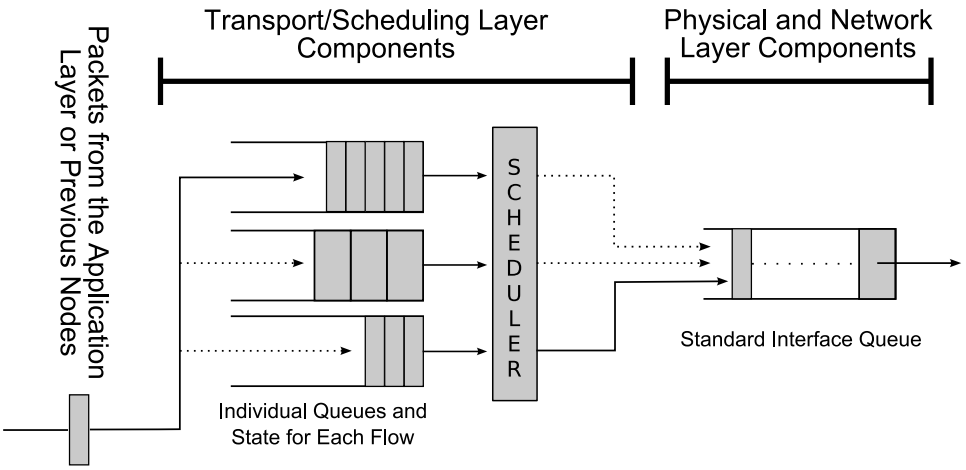

Figure 3.2: *Internal Architecture:* Incoming packets are separated into flow specific queues. An internal scheduler monitors all flows and determines the next packet to be sent.

3.2.1 Internal Architecture

As shown in Figure 3.2, the scheduling layer keeps a separate queue for each flow passing through the node. Doing so allows the node to maintain a more accurate picture of the needs of each flow and helps the node to distribute resources fairly. In order to separate the flows, nodes must be able to distinguish which flow is responsible for each incoming packet. HxH packets assist in this task by including a copy of the sender and receiver's address and port in the packet header. Keeping a copy of this information is necessary, since the destination address in the IP header changes at each hop to ensure that the packet is delivered to each node along the path.

Flows that originate at the node pass their packets directly to the scheduling layer. At intermediate nodes, HxH packets received by the MAC layer are delivered to the scheduling layer. The scheduler checks the HxH header to determine which flow the packets belong to and places them in the appropriate queue. Once queued, the scheduler processes all packets as if they had originated at that node.

When a flow has not been seen before, the scheduler will notice that no state exists when checking the first incoming packet. In such cases, the scheduling layer sets

aside a new queue for the flow and initializes the flow's state to appropriate default values. The packet is then placed into the queue and the flow's state is updated accordingly.

The scheduling layer maintains the flow's queue and state as long as packets continue to be received for the flow. Timers are kept to determine the amount of time that has passed since the last time a packet was received for the flow. If a node becomes disconnected from the flow due to mobility, or the flow becomes inactive, the timers prompt the scheduling layer to remove the state for the flow. Doing so preserves the node's resources. While collecting the results in Chapter 4, state was released whenever flows were inactive for more than two seconds. In practice, the value could be adjusted based on the amount of resources available to the node and the maximum delay expected by the traffic pattern.

3.2.2 Selecting and Monitoring Transmitted Packets

In addition to maintaining state, the scheduling layer is also responsible for determining the order in which packets are sent. To fairly distribute the channel utilization between all flows, the scheduler cycles through all flows in a byte based round robin fashion, similar to existing fair queuing schemes [4].

Once a packet has been selected, the scheduler waits while it is passed down to the network and MAC layers. When the packet is transmitted, the MAC layer uses a callback function to inform the scheduler of the result of the packet transmission. If the transmission was successful, the scheduler selects the next packet to send and the process continues. If the packet was not sent successfully, the scheduler places a copy of the packet back into the flow's queue.

Regardless of whether a transmission was successful or not, the flow's state is updated to indicate the time of the last attempted transmission. For successful transmissions, this value is used to maintain fairness when selecting packets to send.

For unsuccessful transmissions, the update ensures that the scheduler won't repeatedly attempt to send packets from problematic flows without first giving other flows a chance to transmit.

3.2.3 Freezing Flows

In networks with mobility, packets may be delayed if nodes downstream move out of transmission range. In such cases, a new route must be established between the source and destination nodes. In the meantime, intermediate nodes that are aware of the route error freeze their flows in a manner similar to TCP-ELFN [9]. The scheduler ignores frozen flows when determining which packet to send next.

Once a new route is established for a frozen flow, the node will begin receiving new packets. Frozen flows become active again any time a packet corresponding to the flow is received and the node has an available route for the flow. Once the flow is reactivated, the scheduler returns to considering its packets when determining the next packet to send.

Unfortunately, it is possible for frozen flows to receive packets before a new route has been established. In order for a new route to be established, a route error message must be sent to the source. These error messages are handled by multihop wireless routing protocols, and are not a part of HxH. In nearly all cases, the error packets are sent using broadcast packets [3], which do not reserve the channel. Occasionally, the channel is busy when the broadcast packets are sent, causing them to be lost [7].

The process of freezing flows helps to alleviate this problem. Once frozen, flows stop transmitting. If error messages are subsequently lost upstream, the upstream nodes will continue sending packets. When packets arrive for a frozen flow, the scheduler checks to see if a new route has been established. If not, the scheduler causes the routing protocol to *re-broadcast* the error notification. Since the flow is

frozen at the transmitting node, contention is reduced and the chances of the new packet being received upstream are increased. Eventually, the message reaches the source and a route request is made.

3.3 Flow Control

One of the primary limitations hindering the performance of TCP is its inability to distribute bandwidth fairly among nodes as network conditions change rapidly. The HxH packet scheduler helps to reduce the effect of this problem by providing fair queuing. In addition, an effective flow control mechanism is needed that can detect network changes where they occur and adapt quickly to the current conditions of the network.

HxH meets these needs. It is able to rapidly adjust its transmission speed to the available bandwidth at each node, even as other nodes enter and leave the contention area or other flows are created. Additionally, HxH is able to determine appropriate times to send packets without inducing loss. This is particularly important because packet loss in a wireless networks is very costly, since other nodes could have used the wireless medium to transmit data of their own.

To achieve these results, HxH uses a credit-based flow control algorithm at each hop. Standard credit-based algorithms require the source to have a credit for each packet it sends, with the destination explicitly delivering credits to the source. HxH improvemes on this basic credit scheme by listening in on packets sent downstream. As a result, the HxH protocol is able to perform credit-based flow control without explicitly passing credits between nodes.

Figure 3.3 illustrates the operation of the HxH flow control algorithm. Within the scheduler, each flow maintains a pool of credits indicating the amount of buffer space available to that flow at the downstream node. Each time a packet is successfully

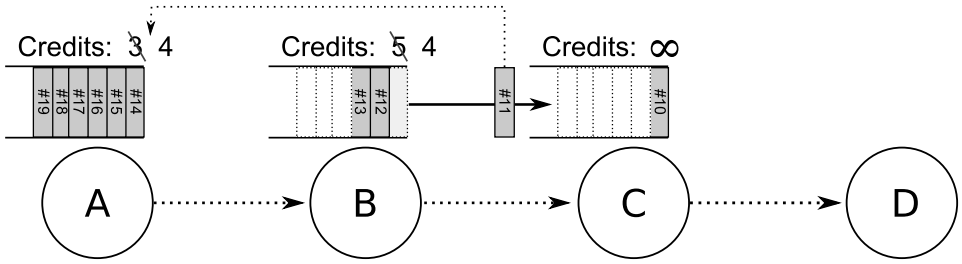

Figure 3.3: *Passive Credit Management:* Each time a node sends a packet, it decrements its credit count. The packet is overheard upstream, allowing nodes to increment their credit count without sending explicit messages.

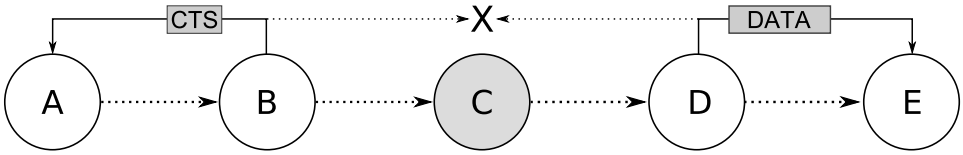

Figure 3.4: *Promiscuous Collisions:* If node B sends a CTS message while node D is sending a data packet, node C will be unable to separate the two messages.

transmitted, the pool is decremented. This process continues as long as the pool indicates that there is space available downstream.

Since neighboring nodes are prevented from transmitting simultaneously, the upstream node can overhear packets sent its downstream neighbors. Each time a packet is sent downstream, the upstream node determines which flow the packet belonged to and adds credits to the flow's pool.

Ideally, this would be the only mechanism needed to provide credit based-flow control. In practice, nodes are sometimes unable to hear every packet sent by their downstream neighbors. As Figure 3.4 demonstrates, collisions can occur at the listening node if nodes further upstream transmit at the same time as nodes downstream. Although both transmissions will be successful, the node attempting to listen in will be unable to distinguish between the two messages.

To account for such circumstances, HxH packets include the current amount of buffer space available for the flow at that node. In cases where occasional packets

are unable to be heard upstream, hearing a single packet will allow the flow to reset its pool to the correct number of credits. Since the transmitting node does not know if the packet transmission will succeed, the value sent represents the amount of buffer space that will be available if the transmission fails.

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.

It is likewise possible that in these extreme cases a node may transmit a packet to a neighbor that has a full queue. To avoid this case, the HxH protocol considers a node's queue to be full at its halfway point. The extra buffer space is used when timers firing upstream cause packets to be sent even though the downstream node has not indicated that there is space in its queue. Even if all the packets stored by an upstream node were to be sent downstream, the queue should still be able to store the packets. Doing so does not greatly affect the upstream node, since it simply causes it to be more conservative in the number of packets sent without the downstream node transmitting.

3.4 Reliability

Another major advantage for HxH is that it eliminates the need to send an explicit acknowledgment for each packet received at the destination. As in the case of flow control, effective schemes for reliability can be created by listening in on data sent downstream.

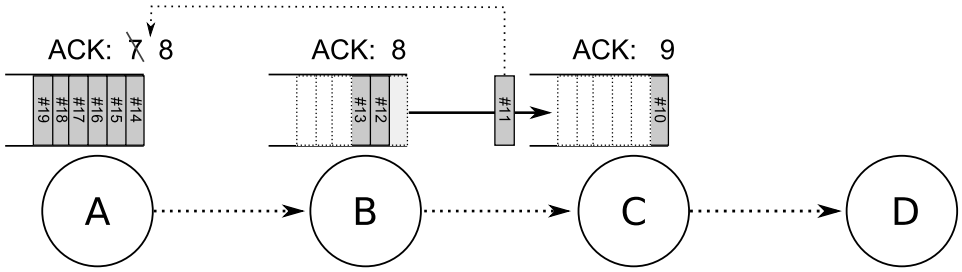

Figure 3.5: *Passive Acknowledgments:* Before transmitting a packet, the node attaches its current ACK value. The ACK value is overheard upstream, allowing ACKs to propagate back to the source.

3.4.1 Passive Acknowledgments

Although the physical layer provides reliability across each hop, additional mechanisms are still needed. If intermediate nodes move, packets that were reliably sent through a portion of the network may never reach the destination. End-to-end reliability is still needed to insure that the destination node receives all the data that is sent.

As part of the state kept for each flow, the scheduling layer maintains a current acknowledgment value for each of the flows passing through it. As shown in Figure 3.5, the ACK value indicates the highest sequence number that the flow's destination has received. For the last hop, this value is updated each time a packet is successfully transmitted to the destination node. Since the 802.11 protocol provides an ACK across a single hop, the second to last node can be sure that the packet was, in fact, received.

To propagate the ACK information upstream, a scheme of passive acknowledgments is used. As in the case of promiscuous flow control, ACK data flows upstream on data moving downstream. Before transmitting, each node modifies a small part of the packet header to indicate its current ACK value. The upstream node listens promiscuously as the packet is sent downstream and updates its own ACK values.

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 acknowledgment packet were received. HxH processes acknowledgments using the same error control techniques as TCP.

3.4.2 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 there is no more data moving through the flow. To handle these cases, HxH uses a bit is in its header to specify that an explicit acknowledgment is requested. Since it is reasonable to assume that the application layer will provide data to the transport layer whenever it is available, the source can set this bit whenever the transport layer's buffers are empty.

Whenever the source requests an explicit ACK, the destination will send one upstream, subject to a rate limit. When forwarding explicit acknowledgments, intermediate nodes update their ACK state in the same way they would for a passive ACK.

3.4.3 Negative Acknowledgments

Even with a credit-based transport protocol, packet loss can still occur if nodes move, taking their queued packets with them. Even in networks without mobility, packet loss can occur if timers allow the flow control algorithm to repeatedly transmit to a downstream node even though it does not have sufficient buffer space.

To cope with packet loss, HxH incorporates negative acknowledgments. As a node receives packets, it checks the sequence number against those that it has already received. If it detects a gap, the node generates a NACK packet that it

sends upstream. Unlike traditional end-to-end protocols, the HxH protocol allows the NACK to be formed at intermediate nodes within the flow, allowing errors to be detected and handled as soon as possible.

To prevent downstream nodes from repeating NACKs, the node initiating a NACK includes in subsequent data packets a range of sequence numbers that should not be NACKed. The flow's destination monitors these NACK values and keeps a list of packets that are missing. If too much time passes before the missing packets are received, the destination retransmits the NACK, assuming that the NACK packet was lost. As with explicit acknowledgments, NACK packets moving upstream carry with them current ACK values and buffer states that intermediate nodes can use to augment data gathered promiscuously.

3.4.4 Comparison to Epoch-Based Reliability

Existing protocols for ad hoc networks have suggested a tradeoff between per-packet acknowledgments and epochs. By sending an acknowledgment for every packet received, TCP is able to respond to congestion within one round trip time. The creators of ATP criticize the overhead induced by per-packet acknowledgments and limit feedback to once per epoch [24]. Epochs reduce overhead, but increase the delay before missing packets are detected and retransmitted. As a result, receivers must maintain much larger buffers or drop packets in order to ensure that data is delivered to the application layer in the correct order. Likewise, the delay increases the length of the feedback loop between destination and source. When conditions within the network become undesirable, the source will not be informed until the next epoch.

Passive acknowledgments provide an excellent compromise between per-packet acknowledgments and epochs. Like epoch-based schemes, passive acknowledgments reduce the number of acknowledgment packets that must be sent upstream, reserving the channel along the way. Yet, like per-packet schemes, the flow of ACK data is

continuous, allowing missing packets to be detected quickly and retransmitted sooner than they would be in an epoch-based scheme.

3.5 Resource Considerations

HxH differs significantly from traditional transport protocols. As noted in Chapter 2, research in multi-hop wireless networks has been dominated by end-to-end protocols. This may seem surprising, due to the success of hop-by-hop protocols in ATM and high speed packet switching networks. Nonetheless, it would be unwise to implement a hop-by-hop protocol without considering its drawbacks.

The primary criticism against hop-by-hop protocols is the level of resources required by intermediate nodes. In order to implement a hop-by-hop protocol, nodes must keep state about each of the flows passing through them; otherwise, they would be unable to control the behavior of individual flows. End-to-end protocols avoid this need by keeping complexity at the edges of the network. This allows intermediate links to be as efficient as possible. The code size of protocols within the network is smaller, allowing routers to be simpler and require less processing power. Research has indicated that end-to-end designs should be used unless there is a potential for significant performance gains [23].

In multi-hop wireless networks, the tradeoffs are justified. The overhead required by such techniques is more manageable in wireless networks since, unlike the Internet, they tend to be relatively small. Comparatively few flows are sent through the network and link speeds are significantly lower than in the Internet. Likewise, intermediate nodes can be the same equipment used at the edges of the network, allowing for more processing power and memory usage at interior nodes. These factors combine to make hop-by-hop protocols feasible and effective. Recent research in multi-hop wireless networks has been successful by making similar tradeoffs [11, 21, 24, 28, 29, 30].

Chapter 4

Results

This chapter presents the results of simulations used to verify the effectiveness of HxH in multi-hop wireless networks. The simulations were conducted in *ns-2* using an implementation of HxH developed as part of this thesis. The simulations test a wide variety of scenarios, but can easily be classified into two groups. First, simulations were used to verify the effectiveness of HxH. Second, simulations were used to compare HxH against other existing protocols.

Unless otherwise stated, the simulations in this chapter use 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. As computed by Gast, the capacity of the network has an upper limit of 8.8 Mbps across a single hop [8]. The packet size is 1024 bytes. Ad hoc on-demand distance vector (AODV) routing [22] is used as the wireless routing protocol. Throughput graphs are created using a one-second sliding window, plotted every 100 milliseconds. Any exceptions to these standards are described in the text.

4.1 HxH Evaluation

4.1.1 Packet Scheduling and Flow Control

First, the HxH protocol was tested in the control topology depicted in Figure 4.1. This simple topology was used to verify that the scheduler could create and dispose

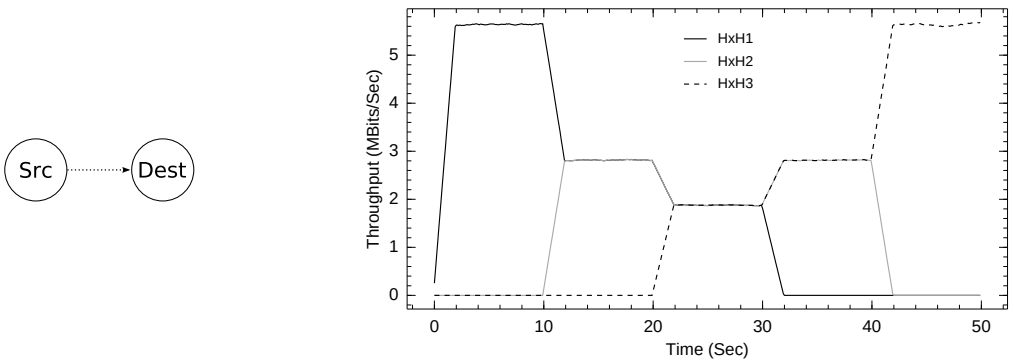

Figure 4.1: *Control Topology (Left):* The simplest case for a hop-by-hop protocol is a single pair of nodes. *Results (Right):* As expected, the scheduler is able to distribute bandwidth between various flows traversing a node.

of state properly and that it would evenly distribute link bandwidth between multiple flows traversing the same node.

In the control topology, three flows were initiated across the link, with one flow starting every 10 seconds. Each flow transmitted data for 30 seconds, the terminated its connection. As expected, the scheduler is able to correctly allocate bandwidth among all flows. Likewise, state was correctly created and freed at the appropriate times.

To ensure that the same results could be expected when extending to multiple hops, the HxH protocol was tested next in the five-node chain topology shown in Figure 4.2. As before, three flows were started, each 10 seconds apart. Each flow continued transmitting for 30 seconds before terminating its connection.

It should be expected that the throughput achieved in the multi-hop case will be even lower than in the single hop case. In a chain topology, the transmission of a single node can prevent nodes several hops away from being able to reserve the channel. For example, when node C transmits, it first sends an RTS to node D. Node D replies with a CTS, which is heard by node E. Even if it has packets to send, Node E will then wait while a packet is sent from C to D. Upstream from node C, a similar effect occurs. As soon as node C begins transmitting, node B's transmission

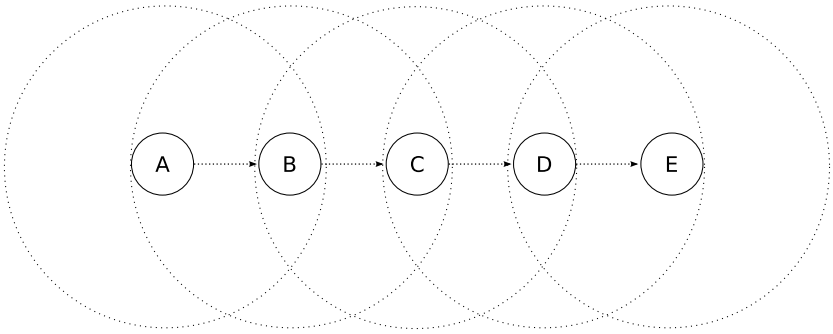

Figure 4.2: *Five Node Chain:* This topology will be used to verify that correct behavior seen across a single hop can be maintained across multiple hops.

channel becomes busy. Even if node A wished to transmit, node B would be unable to distinguish A's packet from the data being sent by node C. Thus, while node C transmits, *none* of the other nodes in the network can send data down the chain. These restrictions, which are a result of using the 802.11 MAC protocol with channel reservations, are well known in the literature [13].

Figure 4.3 shows the results of testing HxH across five nodes. As in the single hop case, the scheduler at each hop is able to evenly distribute the node's bandwidth among the three flows. In addition to receiving lower throughput than in the single hop case, it can also be noted that throughput in the multi-hop case is not quite as smooth as the throughput received across a single hop. This, also, is due to interactions with the 802.11 MAC protocol. Nodes transmit in a random order, with upstream nodes occasionally receiving access to the channel more often than downstream nodes, and vice versa. Because of this, the receiving node's measured throughput fluctuates slightly over time, depending on how smoothly the channel access is distributed amongst all nodes by the MAC protocol.

In addition to maintaining a fair allocation of bandwidth, flow control is also responsible for preventing packet loss across multiple hops. In the chain topology, the credit-based scheme is able to correctly prevent packets from being transmitted

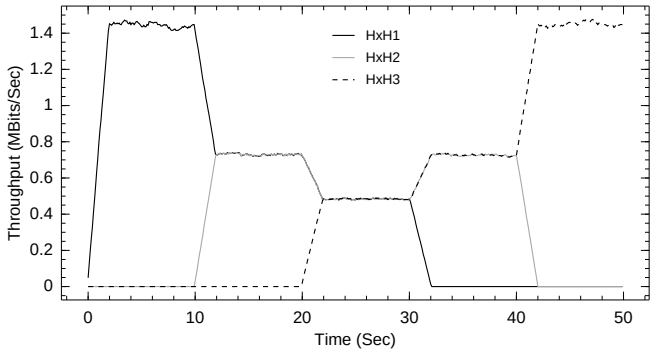

Figure 4.3: *Multi-Hop Results:* Even when HxH is run across multiple hops, the scheduler at each hop is able to evenly distribute bandwidth.

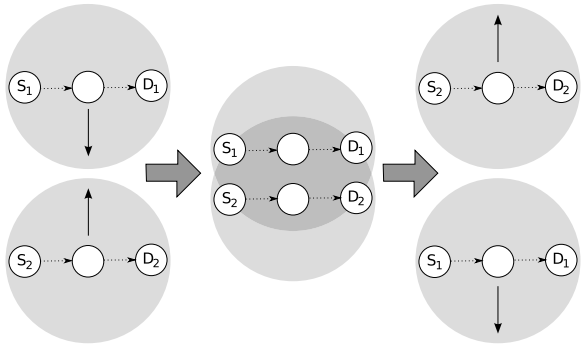

Figure 4.4: *Mobile Topology:* Mobility causes two flows to interact with little warning. As the flows come together, their bandwidth is cut in half. In time, the nodes separate and are able to obtain the entire channel once again.

to nodes with full buffers; no packets are lost. No NACKs are sent and the entire message is received in order.

Since the credit-based flow control scheme was also designed to handle rapidly changing network conditions, HxH was next tested in the topology shown in Figure 4.4. In this mobile topology, two flows are communicating. The flows initially start out far enough from each other that they 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. Due to mobility, the flows see rapid reductions and gains in their attainable throughput.

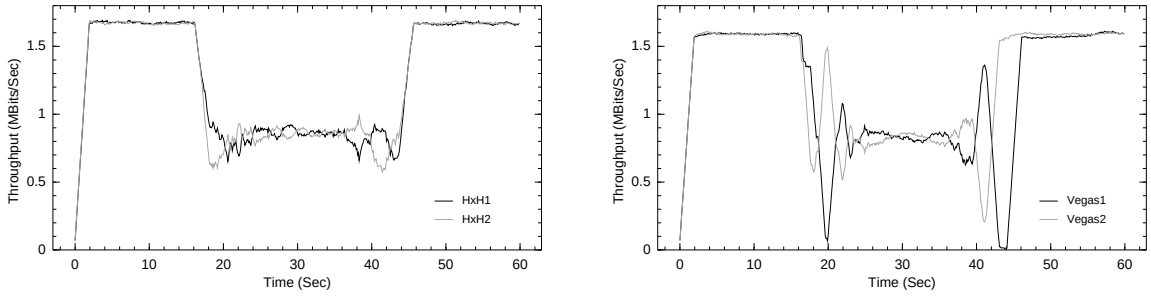

Figure 4.5: *HxH Outperforms End-to-End:* HxH (left) is able to respond to dramatic changes in network conditions faster and with less dramatic dips than end-to-end protocols (right).

Since the primary focus of this test is determining the effectiveness of the hop-by-hop flow control mechanism, passive acknowledgment were disabled. Instead, HxH sent an explicit end-to-end acknowledgment packet upstream for every data packet received at the destination. This behavior mimics TCP's reliability mechanism, ensuring that any benefits seen are solely the result of HxH's flow control algorithm. Figure 4.5 compares the results of HxH to those seen by TCP Vegas, which was the most effective end-to-end protocol in this scenario. As expected, hop-by-hop flow control is able to respond to the changing network conditions much more quickly than end-to-end protocols.

4.1.2 Passive Acknowledgments

The previous scenario can also be used to determine the effect of passive acknowledgments on HxH's performance. Since using passive acknowledgments eliminates the need to reserve the channel for small ACK packets moving upstream, more bandwidth is available for data packets. Figure 4.6 demonstrates the effect of passive acknowledgments on HxH's performance. In this case, using passive acknowledgments allows for a throughput gain of 73%.

As expected, the benefit obtained by passive acknowledgments varies with the speed of the wireless link. As shown in Table 4.1, higher speed links see greater

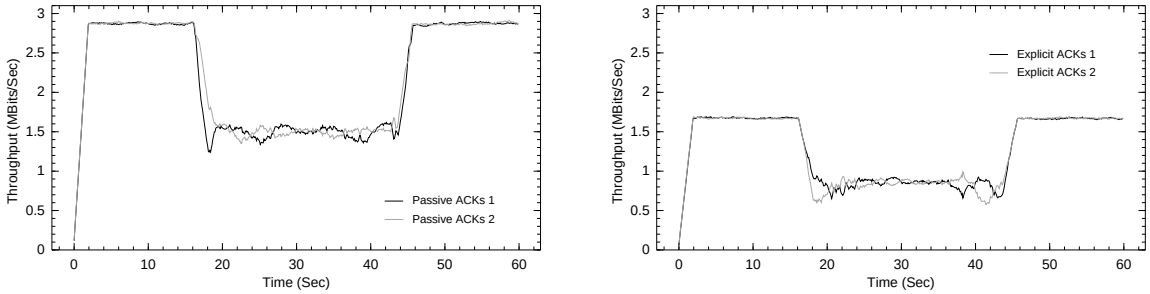

Figure 4.6: *Passive vs Explicit Acknowledgments:* HxH performance is shown here with (left) and without (right) passive acknowledgments. Passive acknowledgments allow for significant performance gains.

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

Table 4.1: *Passive vs Explicit ACKs:* As link speeds increase, the benefit of using passive acknowledgments also increases.

performance improvements from passive acknowledgments than slower links. This occurs because the overhead required to send a packet with channel reservations varies with link speed. These results confirm the calculations from 1.2. Since passive acknowledgments eliminate the need for end-to-end ACK packets, more of the channel is made available for data packets moving downstream. With any link speed the benefits achieved by passive acknowledgments are desirable.

4.1.3 Negative Acknowledgments

Although credit-based flow control prevents packets from being dropped in most circumstances, loss can still occur due to mobility. If intermediate nodes that are buffering packets get separated from the rest of the flow, HxH uses negative acknowledgments to inform the source that retransmissions are needed.

To test the effectiveness of HxH's negative acknowledgments, the topology shown in Figure 4.7 was used. During the simulation, intermediate nodes were repeat-

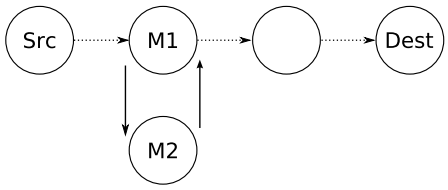

Figure 4.7: *Unreliable Chain:* Mobility was used to test reliable packet delivery. Nodes $M1$ and $M2$ periodically traded places, causing queued packets to be lost as new routes were formed.

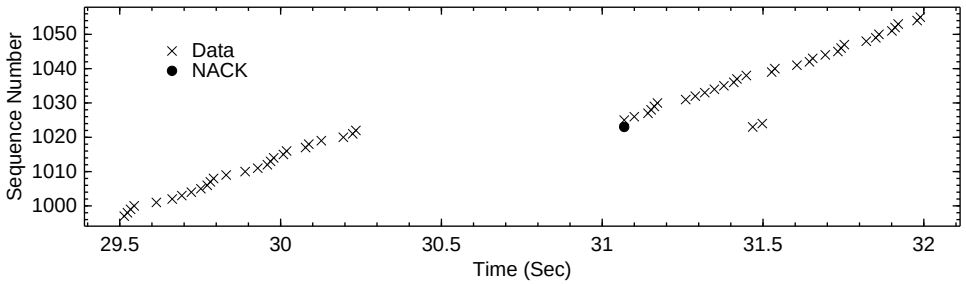

Figure 4.8: *Negative Acknowledgments:* When nodes move, packet loss may occur. After new routes are established, gaps in sequence numbers are NACKed and the packets are re-transmitted.

edly separated from the path to force packet loss. The source continued transmitting as usual, assuming no data has been lost. After new routes were established, the first node to detect a gap in the delivered sequence numbers transmitted a NACK packet upstream.

Figure 4.8 shows one instance of the HxH protocol recovering from mobility-induced packet loss. To prevent an overwhelming number of packets from being displayed in the graph, the simulation was run with a link speed of 1 Mbps. The gap seen in the graph occurs when the flow is frozen as a new route is established. As expected, the second to last node transmits a NACK once the path is reconnected. The sequence numbers continue uninterrupted until the source retransmits the missing packets.

The negative acknowledgment scheme has been further tested using a wide variety of failure frequencies and link speeds. In all cases, NACKs were sufficient to

ensure that the entire message was received. If NACK packets were lost upstream due to node mobility, the destination was able to successfully retransmit the NACK at a later time.

Although NACK packets are useful to ensure that the entire message is delivered successfully, they also have an effect on overall performance. NACK packets have the same drawbacks as the explicit end-to-end acknowledgments sent by TCP. Since the packets are small, they incur a large overhead when reserving the wireless channel. Unlike ACKs in TCP, however, these packets are quite rare while using HxH. Analysis of simulations throughout this chapter indicate that NACK packets make up less than 2% of all packets sent by the HxH protocol. Although NACKs degrade throughput, the performance loss is small in comparison to the gains of passive acknowledgements and hop-by-hop credit-based flow control.

4.2 Comparison to Existing Protocols

To verify that HxH outperforms existing transport protocols in multi-hop wireless networks, it was tested against five existing protocols in a variety of topologies and usage scenarios. The protocols that were used for comparison are TCP-Tahoe, TCP-NewReno, TCP-Sack, TCP-Vegas and ATP. The results of the comparison tests confirm that HxH achieves higher throughput than existing protocols.

4.2.1 Dumbbell Topology

First, the protocol was tested in situations of high congestion. As shown in Figure 4.9, four flows were established across a shared four-hop link, causing a substantial bottleneck. The flows were staggered with one flow starting every ten seconds in order to observe the protocol's behavior as flows adjusted their rates.

The dumbbell topology also helps to test the effect of a problem unique to ad hoc wireless networks. In such networks, node mobility can cause routes to fail,

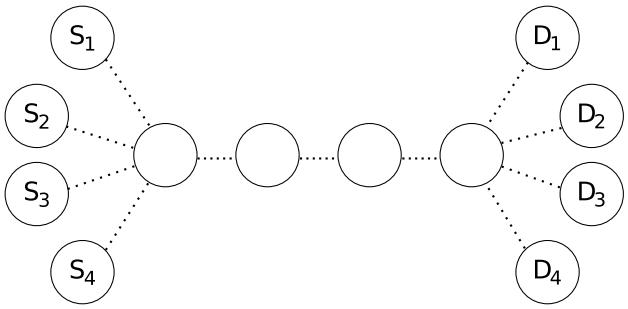

Figure 4.9: *Dumbbell Topology:* This topology simulates networks experiencing heavy congestion. Nodes clustered in two groups communicate over four highly congested nodes.

requiring the source to establish a new path to the destination before transmitting. To detect these failures, routing protocols observe the number of RTS requests needed before a corresponding CTS message is received. If seven consecutive RTS messages are sent without hearing a CTS from the node downstream, the routing protocol assumes the node has moved.

Unfortunately, heavy wireless contention can cause routing protocols to believe that nodes have moved when they have not. Figure 4.10 shows how this can occur. Whenever node C transmits to node D, node B's channel is busy. If node A attempts to request the channel to transmit to node B, it will receive no response. When nodes A and C are sending at the same time, node B is unable to understand either message. If the problem persists over several consecutive RTS attempts, the node requesting the channel will report a route failure, even though the route is intact.

Figure 4.11 presents the results of running each of the test protocols in the dumbbell topology. The results highlight many of the benefits of using HxH. It is immediately noticeable that HxH achieves much higher throughputs than all other protocols except ATP. The use of passive ACKs eliminates the need to reserve the channel for acknowledgment packets, thus freeing the bandwidth for data packets moving downstream. ATP is able to achieve similar throughputs through the use of epoch-based ACKs, but cannot maintain a stable rate.

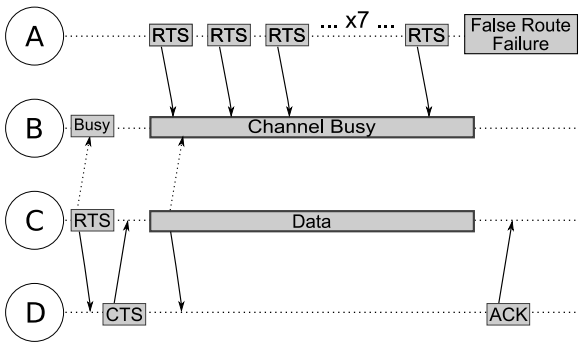

Figure 4.10: *False Route Failure:* Node B can not hear node A's RTS messages when node C is transmitting. After seven failed attempts, node A incorrectly assumes node B has moved.

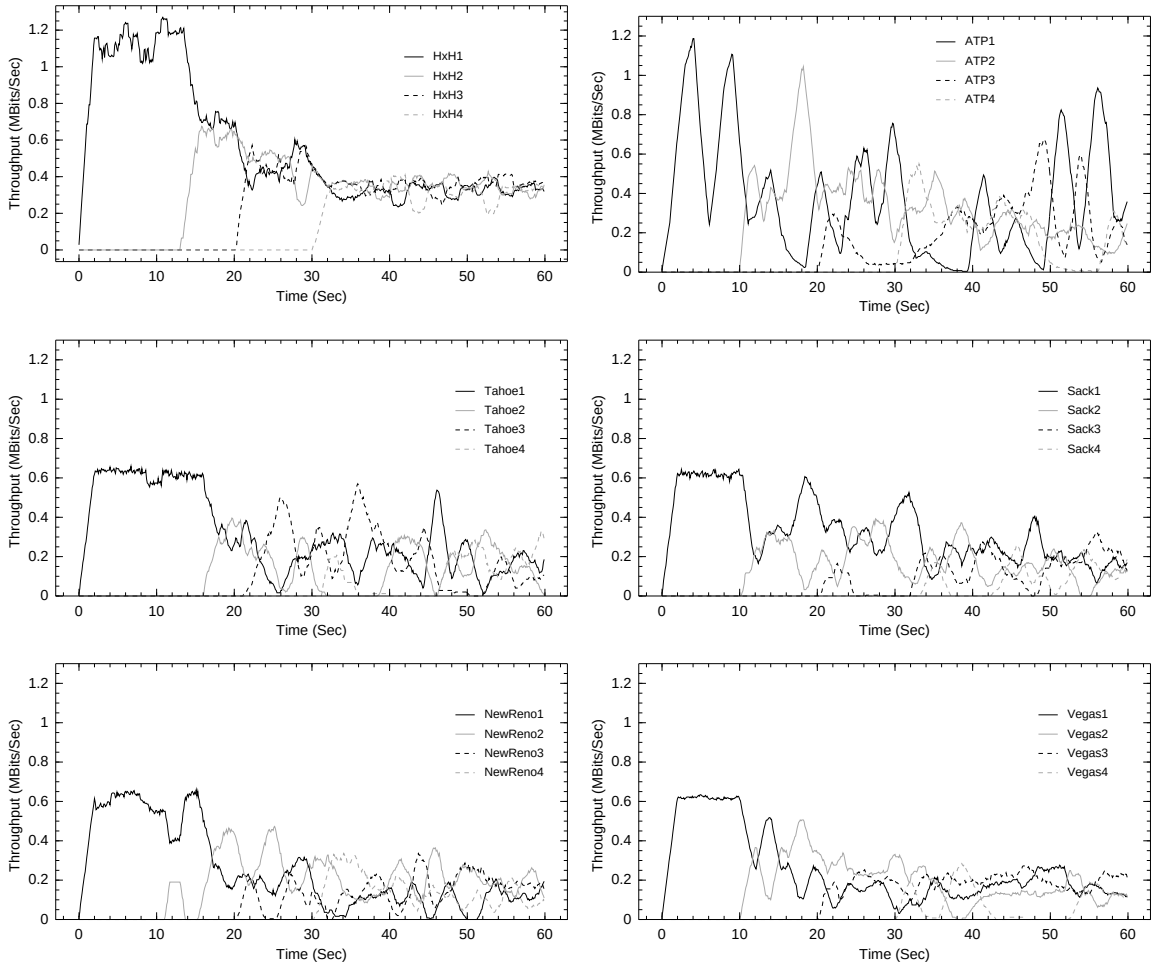

Figure 4.11: *Dumbbell Results:* HxH (top-left), ATP (top-right), Tahoe (middle-left), Sack (middle-right), NewReno (bottom-left) and Vegas (bottom-right) running in the dumbbell topology. HxH outperforms the other protocols in overall throughput and fairness among flows.

	HxH	ATP	Tahoe	Sack	NewReno	Vegas
Flow 1	1.21	1.04	0.75	0.83	0.38	0.61
Flow 2	1.33	0.93	0.47	0.53	0.72	0.50
Flow 3	1.31	0.99	0.64	0.57	0.59	0.73
Flow 4	1.20	0.82	0.43	0.33	0.57	0.32
Total	5.05	3.79	2.29	2.27	2.26	2.17
Fairness	0.998	0.993	0.951	0.910	0.956	0.928

Table 4.2: Summary of Performance in a Dumbbell (Megabytes Received)

In addition to achieving higher rates, HxH is also performs better than other protocols as new flows begin contending for the wireless channel. In all protocols, the flows suffer from contention-induced (false) route failures and must deal with the failures. These route failures can be observed as sudden dips in the throughputs of the various flows. Versions of TCP interpret the drop in bandwidth and the increased end-to-end delay as an indication to slow down, but overcompensate. The HxH protocol responds to the route failures more appropriately. The flows simply continue requesting the channel, but receive reduced throughputs as the MAC layer shares bandwidth between the flows. As a result, the dips in the HxH are less severe and are handled more quickly.

By responding more quickly to changing conditions and using a packet scheduler, HxH is able to achieve a more fair allocation of bandwidth than other protocols. The Jain's fairness equation (Equation 4.1) was used to determine how fairly the available bandwidth was distributed among the four flows. The results are summarized in Table 4.2 and confirm that HxH is the most fair of all the protocols tested. The data shown in the table represents only the last 30 seconds of the simulation, when all four flows are actively participating.

$$fairness = \frac{(\sum x_i)^2}{(n \cdot \sum x_i^2)} \tag{4.1}$$

The throughput graphs of the various protocols also demonstrate weaknesses within the 802.11 MAC protocol. With HxH, Tahoe and NewReno, the second flow does not initially begin transmitting until several seconds after it should. This delay is caused by the loss of an AODV route request message. As noted by [7], RTS/CTS traffic can prevent broadcast packets such as the AODV route request from being delivered.

Unlike many of the TCP protocols, HxH's measured throughput is initially a bit jagged. This occurs because HxH relies entirely on the MAC protocol to determine when packets should be sent. Although the 802.11 MAC is meant to be globally fair among nodes throughout time, it has been shown to be unfair on a short-term basis [27]. As a result, nodes along a path are not granted access to the channel on a consistent basis. Sometimes upstream nodes receive the channel more frequently than downstream nodes, and vice versa. The random assignment of the channel is largely responsible for the behavior seen in the first few seconds of the graph.

4.2.2 Mobile Topology

Since wireless nodes are potentially mobile, it is necessary to determine how transport protocols respond to mobility-induced route failures. The protocols must adapt to failed routes and continue communicating once new routes are established. Figure 4.12 presents the test scenario used to evaluate each protocol under varying conditions of mobility. The source and destination nodes moved along the edges of the lattice, inducing route failures and requiring new routes to be formed.

The simulation was repeated with varying numbers of forced route failures to determine how increasing mobility affects each protocol's overall performance. In order to keep the results from multiple simulations comparable, all simulations were run for the same amount of time (60 seconds). Increased numbers of route failures

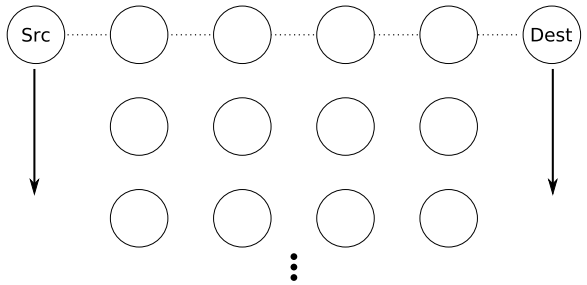

Figure 4.12: *Mobile Lattice:* Moving the source and destination along a fixed lattice forces each protocol to respond to route failures.

were induced by increasing the speed with which the source and destination nodes moved along the lattice.

Figure 4.13 summarizes the performance of each protocol in the mobile lattice. When the number of route failures is zero, the performance of each protocol is similar to the values seen in the first ten seconds of the dumbbell topology. As before, HxH is able to achieve a higher initial throughput and maintain the rate due to passive acknowledgments and its hop-by-hop flow 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 rate at which each protocol's performance degrades varies. Vegas, Tahoe, NewReno and Sack each suffer a drop of nearly 50% as the number of route failures is increased from zero to 60. Under the same conditions, HxH only loses 8.2% of its total throughput. This performance gain occurs mostly due to HxH's understanding that loss can be caused by mobility as well as congestion. The various versions of TCP were designed with the assumption that *all* loss is due to congestion [10]. Their corresponding behavior hinders them in mobile networks.

Like HxH, ATP was designed with mobility in mind. As a result, it's throughput does not decrease significantly as the number of route failures increases. Unfortunately, ATP also suffers from instability in flows traversing more than four nodes. Since ATP restricts rate-based decisions to once per second, any network changes

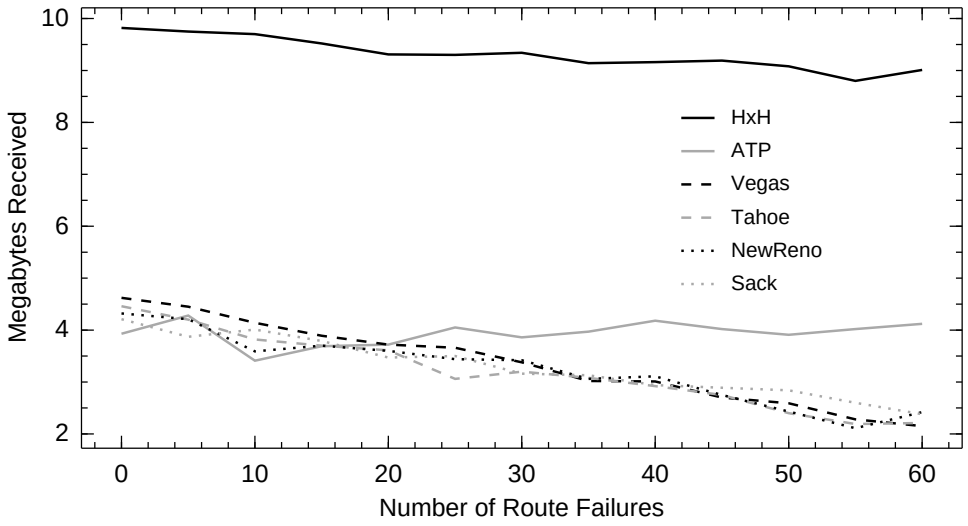

Figure 4.13: *Coping with Mobility:* Throughput decreases as the number of route failures increases. HxH throughput drops at a slower rate than existing protocols as mobility increases.

that occur during the epoch cause queuing to occur. Eventually, ATP detects the increasing delay and lowers its transmission rate. Frequently, ATP overcompensates and chooses rates that are much lower than necessary. Once the flow's queues are back to normal, the process repeats. As a result, route failures in mobile networks actually *help* ATP. Since each route failure gives ATP a fresh start with a new set of nodes, the effect of its instability is minimized. With a large enough number of route failures, ATP performance degrades at a rate similar to that seen by HxH.

4.2.3 Grid Topology

The dumbbell and mobile lattice topologies are helpful to determine the effects of specific network conditions. Unfortunately, neither is truly representative of the types of topologies one would expect to see in a multi-hop wireless mesh network. In many cases such networks may be planned beforehand, with nodes placed strategically in fixed locations to maximize performance.

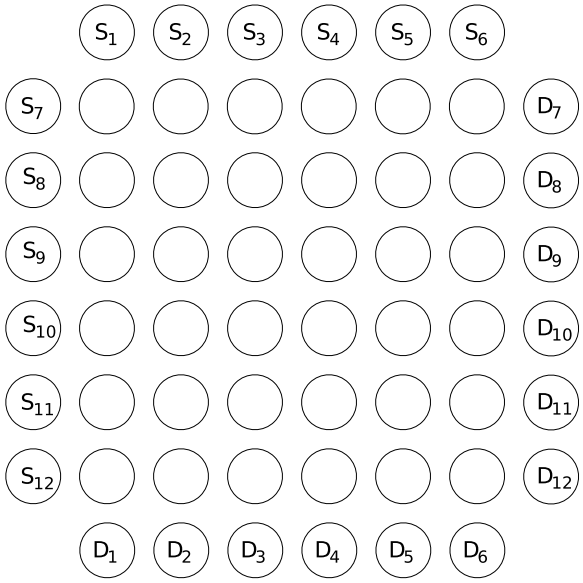

Figure 4.14: *Grid Topology:* Twelve flows communicate using a six-by-six lattice as an infrastructure for their communication.

The grid topology shown in Figure 4.14 was used to better model a planned mesh network. In this topology, a six-by-six grid provides the framework across which twelve flows communicate. Since the simulation is meant to model a planned topology, no ad hoc routing protocol was used. Instead, each flow was given a route through the grid which it maintained thoughout the entire simulation. All false route failures were suppressed.

Figure 4.15 presents the number of bytes received by each flow for each protocol during a 60 second period. Table 4.3 summarizes the results, showing the sum total of all flows during the simulation. HxH and ATP both achieve higher throughputs due to the smaller number of explicit ACKs they send. Vegas, NewReno, Tahoe and Sack all achieve similar collective throughputs.

Interestingly, *none* of the protocols are able to achieve a fair allocation of bandwidth across all twelve flows. Regardless of the protocol being used, some flows receive significantly higher throughputs than others. Although the transport protocols themselves can worsen this problem, the unfairness seen is not exclusively a

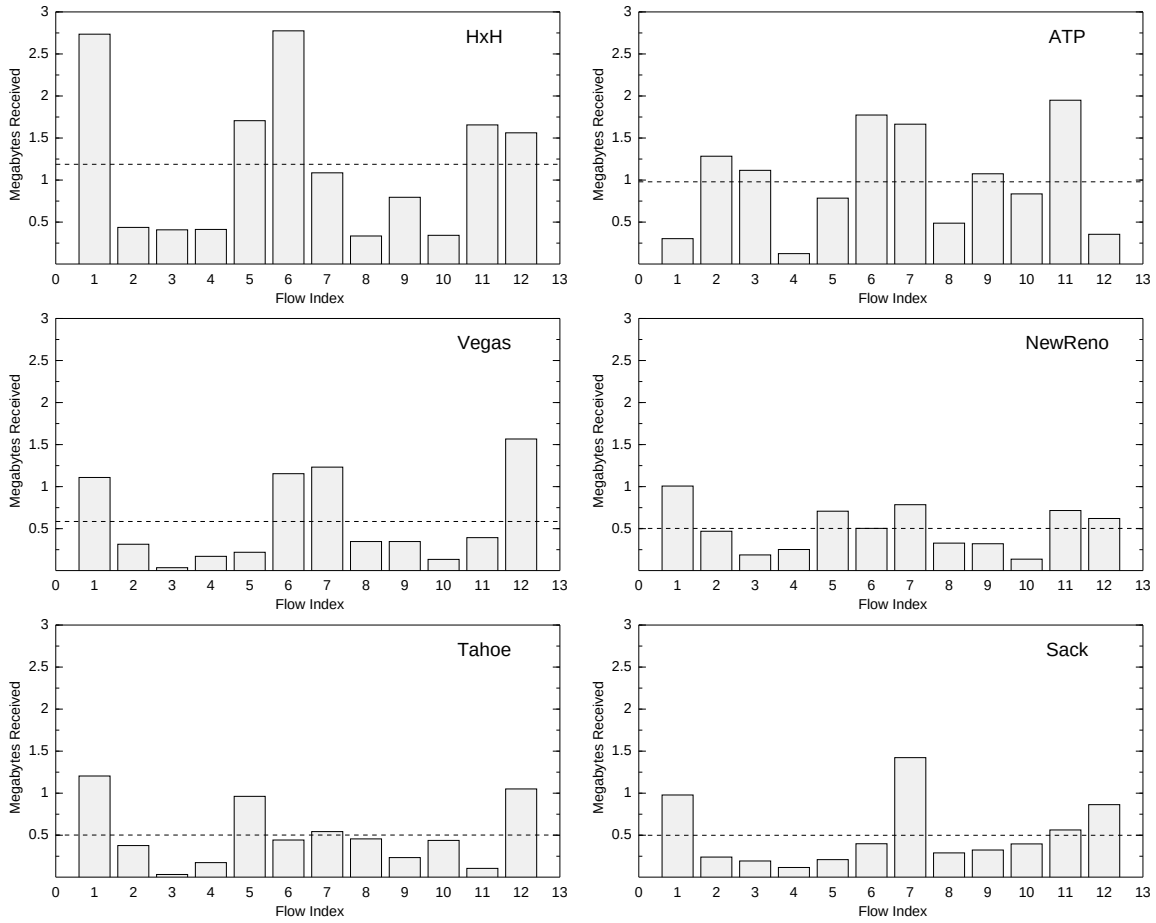

Figure 4.15: *Grid Results:* HxH (top-left), ATP (top-right), Vegas (middle-left), NewReno (middle-right), Tahoe (bottom-left) and Sack (bottom-right) running in the planned grid topology. The MAC layer allows flows on the edge of the grid to access the channel more frequently than flows in the middle of the grid.

Protocol	**MBytes Received**
HxH	14.25
ATP	11.75
Vegas	7.02
NewReno	6.03
Tahoe	6.02
Sack	5.99

Table 4.3: Summary of Performance in a Planned Grid Topology

transport layer issue; the MAC layer and the topology are itself are primarily responsible. Due to the shape of the topology, some flows compete with fewer flows than others.

The 802.11 MAC protocol makes an attempt to allocate bandwidth fairly among nodes competing in the same contention range. In cases where nodes are experiencing similar contention on the wireless channel, it is successful. In cases where some nodes experience much higher contention than others, however, the nodes with fewer neighbors to compete with achieve higher throughputs. This problem is well known [28], and is the reason why flows on the edge of the grid (flows 1, 6, 7 and 12) and their neighbors tend to receive higher throughputs than flows in the center of the grid.

Although the transport layer cannot entirely fix the problem, it can help to some degree. The use of a packet scheduler in the HxH protocol helps flows with similar contention ranges to achieve similar access to the channel. This can be seen by observing how closely the throughputs of interior flows match each other in the HxH case. Likewise, flows on the edge achieve similar results, as do their neighbors. Without some care for fairness, existing protocols give significantly less bandwidth to some flows than others, even though they are experiencing similar conditions of wireless contention.

4.2.4 Random Topologies

To ensure that the benefits seen by HxH are not the result of well selected topologies, the protocol was also tested in a set of 50 randomly generated scenarios. In each of the random scenarios, 200 mobile wireless nodes were scattered randomly throughout four square kilometers ($2Km \times 2Km$) of terrain. The density of nodes within the terrain is intended to maintain a single connected network without partitions. In each scenario, five flows were created among the nodes. The flows communicated

	HxH	Vegas	NewReno	Sack	Tahoe	ATP
MBits/Second (Avg)	2.676	1.488	1.423	1.391	1.385	1.069
MBytes Received (Avg)	20.07	11.16	10.67	10.43	10.39	8.02
Standard Deviation	9.57	4.40	5.11	4.75	4.99	2.45

Table 4.4: Summary of Performance in Random Topologies (Megabytes Received)

to the best of their ability over the course of 60 seconds. Meanwhile, nodes moved throughout the terrain using a random-waypoint model. After waiting a random period of time between five and 25 seconds long, nodes would move to a new random location. They would then choose a new delay and the process would repeat.

The mean and standard deviation of the observed throughputs for all six protocols are summarized in Table 4.4. As expected, the HxH protocol outperforms all others. It's improvement is statistically significant[1] ($p < 0.0001$). Surprisingly, ATP performs significantly *worse* than the other protocols ($p < 0.0001$). As observed in previous sections, the instability of ATP in long flows outweighs its ability to increase throughput by sending ACKs only once per epoch.

Among the various versions of TCP, there is not a statistically significant difference between protocols. It is important to note, however, that these results relate to throughput only. Although their throughputs are similar, the fairness among flows and smoothness of the achieved throughputs are dramatically different.

[1]Statistical significance was verified using a two-sample t-test between all pairs of protocols.

Chapter 5

Conclusions

This thesis has detailed the design of HxH, a new transport protocol for multi-hop wireless networks. HxH was created to overcome two weaknesses observed in TCP performance:

1. Throughput is not fairly allocated among wireless nodes.
2. The use of explicit acknowledgment (ACK) packets incurs an unreasonably high overhead.

To overcome the unfairness of TCP, HxH uses a credit-based flow control scheme at each hop. By allowing each node along the path to participate, HxH has a better view of network conditions. Even as rapid changes occur within the network, the protocol can detect and respond to the new conditions where they occur.

Fairness is also improved through the use of a packet scheduler at each node. With packets separated into flow-specific queues, the protocol is able to ensure that each flow at a particular node is given an equal opportunity to access the wireless channel.

As demonstrated through simulation, HxH is able to achieve a higher fairness than existing protocols when flows are experiencing similar degrees of wireless contention. When the contention seen by each flow is not equal, flows with lower contention are able to access the channel more frequently. This occurs because band-

width allocated to each node by the 802.11 MAC protocol is not fairly distributed [28].

The second problem addressed by HxH is reducing overhead incurred by sending explicit acknowledgment packets. To do so, HxH uses a scheme of passive acknowledgments. Passive acknowledgments eliminate the need to send ACK packets upstream, which frees up the wireless channel for more data to be sent.

Simulations confirm that passive acknowledgments are able to significantly improve the total throughput achieved by flows in multi-hop wireless networks. The benefit received increases with link speed, but is desirable even when using slower links.

HxH dramatically improves performance in networks with or without mobility. The use of passive ACKs increases the total throughput achieved by each flow. The hop-by-hop nature of the protocol allows it to respond more appropriately to rapid changes, improving fairness and the overall shape of the observed throughput graphs and histograms. These results have been submitted for publication in ACM MobiCom and are under review from the program committee.

5.1 Open Issues and Future Work

The development of HxH has suggested the need for further research in two main areas: improving the 802.11 MAC protocol and the development of a rate-based HxH protocol.

5.1.1 Improving the 802.11 MAC

Throughout the course of this thesis, several problems with the existing 802.11 MAC protocol have been outlined. Further research is needed to find ways of improving the MAC layer. The resulting benefits would improve the performance of all existing

transport protocols, not just HxH. Future research could focus on any of the following problems:

1. **False Route Failures:** At present, routing protocols are unable to distinguish between mobility-induced route failures and failed transmissions caused by high contention. As a result, nodes frequently request new routes even though they are not necessary. Research is needed to develop better cross-layer interactions between the MAC layer and routing protocols to eliminate false route failures.
2. **Lost Broadcast Packets:** The RTS/CTS messages provided by the 802.11 MAC reduce the likelihood that messages will be lost due to interference. At present, broadcast packets are unable to use RTS/CTS protection, because there is not a single destination node with whom to request the channel. Even in networks using RTS/CTS protection, broadcast packets are frequently lost. The result is particularly damaging when ad hoc routing messages are lost. The lost message must be flooded through the network multiple times, tying up the bandwidth of many nodes. Meanwhile, the source is unable to transmit until a route reply is successfully received. This problem could be alleviated by providing mechanisms for reliable broadcast in wireless networks. Some initial work has been done in this area [16, 25], but more research is necessary.
3. **Fairness Among Nodes:** Although the MAC layer for 802.11 was designed with fairness in mind, it falls short of its desired goals. Nodes on the fringes of networks are able to access the channel much more often than nodes deep within the network. In severe cases, this unfairness can lead to starvation of nodes [28]. This problem also contributes to the occurrence of false route failures, since nodes on the edge of the network can access the channel more frequently than their downstream neighbors will be able to allow them to send. Research is needed to find ways of fairly allocating bandwidth at the physical layer and to help nodes detect neighbors experiencing heavy contention.

5.1.2 Rate-Based HxH

Counterintuitively, research during the development of HxH has suggested that even greater throughput may be achieved in wireless networks by *limiting* the rate at which packets are sent. In the credit-based scheme, nodes always try to access the channel as long as there is space available downstream. By attempting to access the channel whenever possible, the protocol occasionally requests the channel while it is in use downstream, and must wait. In the worst case, this can lead to false route failures. More frequently, it leads to the MAC protocol entering a backoff state, in which the delay before requesting the channel in the future is increased.

Rather than sending at every possible opportunity, it would be better to send packets only when the channel is truly available. Doing so would require that the protocol wait occasionally, even if it perceives an idle channel. The *ideal* rate at which to send would be the highest possible rate that still allows every packet to be successfully sent on its first attempt. By 'slowing down' to this ideal rate, protocols could avoid the possibility of false route failures and actually see increased throughputs, since the MAC layer would not be forced to back off.

The benefit of a rate-based protocol can already be seen in the differences observed between TCP-Vegas and its window-based counterparts in multi-hop wireless networks. By limiting its rate, Vegas sends at a slightly slower maximum speed than the other protocols, but is able to maintain the rate over much longer periods of time. Similar results could be achieved with a rate-based hop-by-hop protocol.

Figure 5.1 demonstrates the potential gains of such a protocol. In the graph, a single flow was initiated across a ten hop (11 node) chain. Like TCP-Sack, the credit-based HxH protocol suffers from occasional drops in throughput as a result of its constant requests to access the channel. Like Vegas, an incomplete rate-limited version of the HxH protocol was able to achieve data rates that are much higher and smoother than its more aggressive counterpart.

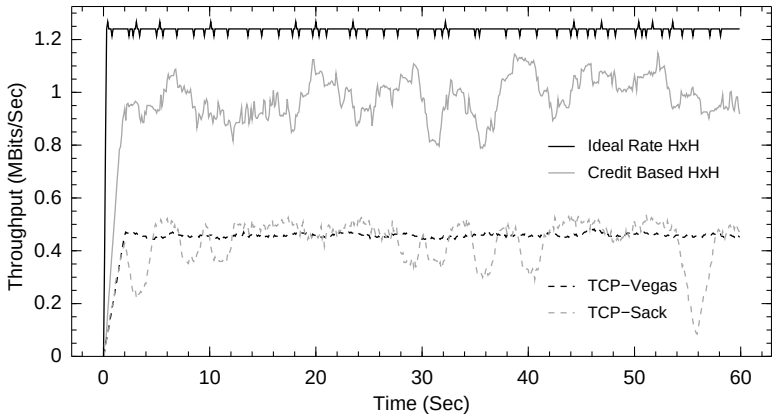

Figure 5.1: *Rate vs Credit Based Schemes:* Whether using hop-by-hop or end-to-end designs, rate based protocols outperform protocols that attempt to access the channel more aggressively.

Initial research into the development of a rate-based protocol has shown it to be an extremely difficult challenge. The ideal rate at which the protocol should send is topology dependent. Rates that work in one topology will fail in another. Thus, the protocol must be able to determine the rate as it is running. This becomes extremely complicated because nodes may be moving at the same time the rate detection is taking place, causing the ideal rate to be moving target.

Readers are encouraged to extend this work by refining a method for choosing appropriate rates in multi-hop wireless networks.

Bibliography

[1] H. Balakrishnan, V. N. Padmanabhan, S. Seshan, and R. H. Katz. A Comparison of Mechanisms for Improving TCP Performance over Wireless Links. *IEEE/ACM Transactions on Networking*, 5(6):756–769, 1997.

[2] L. S. Brakmo, S. W. O'Malley, and L. L. Peterson. TCP Vegas: New Techniques for Congestion Detection and Avoidance. In *ACM SIGCOMM*, pages 24–35. ACM Press, 1994.

[3] J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. Hu, and J. Jetcheva. A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols. In *ACM MobiCom*, pages 85–97. ACM Press, 1998.

[4] A. Demers, S. Keshav, and S. Shenker. Analysis and Simulation of a Fair Queueing Algorithm. In *ACM SIGCOMM*, pages 1–12. ACM Press, 1989.

[5] S. M. ElRakabawy, A. Klemm, and C. Lindemann. TCP with Adaptive Pacing for Multihop Wireless Networks. In *ACM MobiHoc*, pages 288–299. ACM Press, 2005.

[6] S. M. ElRakabawy, C. Lindemann, and M. K. Vernon. Improving TCP Performance for Multihop Wireless Networks. In *IEEE/IFIP DSN*, pages 684–693. IEEE Computer Society, 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] M. S. Gast. *802.11 Wireless Networks: The Definitive Guide*. O'Reilly & Associates, Inc., Sebastopol, CA, USA, 2002.

[9] G. Holland and N. H. Vaidya. Analysis of TCP Performance over Mobile Ad Hoc Networks. In *Mobile Computing and Networking*, pages 219–230, 1999.

[10] V. Jacobson. Congestion Avoidance and Control. In *ACM SIGCOMM*, pages 314–329. ACM Press, 1988.

[11] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Medard, and J. Crowcroft. XORs in the Air: Practical Wireless Network Coding. In *ACM SIGCOMM*, 2006.

[12] H. T. Kung, T. Blackwell, and A. Chapman. Credit-Based Flow Control for ATM Networks: Credit Update Protocol, Adaptive Credit Allocation and Statistical Multiplexing. In *ACM SIGCOMM*, pages 101–114. ACM Press, 1994.

[13] J. Li, C. Blake, D. Couto, H. I. Lee, and R. Morris. Capacity of Ad Hoc Wireless Networks. In *ACM MobiCom*, pages 61–69. ACM Press, 2001.

[14] J.-S. Li and C.-W. Ma. Improving Fairness of TCP Vegas. *International Journal of Network Management*, 15(1):3–10, 2005.

[15] 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.

[16] W. Lou and J. Wu. Double-Covered Broadcast (DCB): A Simple Reliable Broadcast Algorithm in MANETs. In *IEEE INFOCOM*, 2004.

[17] S. Mascolo, C. Casetti, M. Gerla, M. Y. Sanadidi, and R. Wang. TCP Westwood: Bandwidth Estimation for Enhanced Transport over Wireless Links. In *ACM MobiCom*, pages 287–297. ACM Press, 2001.

[18] P. P. Mishra and H. Kanakia. A Hop by Hop Rate-Based Congestion Control Scheme. In *ACM SIGCOMM*, pages 112–123. ACM Press, 1992.

[19] K. Nahm, A. Helmy, and C.-C. J. Kuo. TCP over Multihop 802.11 Networks: Issues and Performance Enhancement. In *ACM MobiHoc*, pages 277–287. ACM Press, 2005.

[20] C. Özveren, R. Simcoe, and G. Varghese. Reliable and Efficient Hop-by-Hop Flow Control. In *ACM SIGCOMM*, pages 89–100. ACM Press, 1994.

[21] M. Pandey, R. Pack, L. Wang, Q. Duan, and D. Zappala. To Repair or not to Repair: Helping Ad Hoc Routing Protocols to Distinguish Mobility From Congestion. In *IEEE INFOCOM (to appear)*, 2007.

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

[23] J. H. Saltzer, D. P. Reed, and D. D. Clark. End-to-End Arguments in System Design. *ACM Transactions on Computer Systems*, 2(4):277–288, 1984.

[24] K. Sundaresan, V. Anantharaman, H.-Y. Hsieh, and R. Sivakumar. ATP: a Reliable Transport Protocol for Ad-Hoc Networks. In *ACM MobiHoc*, pages 64–75. ACM Press, 2003.

[25] K. Tang and M. Gerla. MAC Reliable Broadcast in Ad Hoc Networks. In *IEEE MILCOM*, 2001.

[26] F. Wang and Y. Zhang. Improving TCP Performance over Mobile Ad-Hoc Networks with Out-of-Order Detection and Response. In *ACM MobiHoc*, pages 217–225. ACM Press, 2002.

[27] 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.

[28] K. Xu, M. Gerla, L. Qi, and Y. Shu. Enhancing TCP Fairness in Ad Hoc Wireless Networks Using Neighborhood RED. In *ACM MobiCom*, pages 16–28. ACM Press, 2003.

[29] Y. Yi and S. Shakkottai. Hop-by-Hop Congestion Control over a Wireless Multi-Hop Network. In *IEEE INFOCOM*, volume 4, pages 2548–2558, 2004.

[30] X. Yu. Improving TCP Performance over Mobile Ad Hoc Networks by Exploiting Cross-layer Information Awareness. In *ACM MobiCom*, pages 231–244. ACM Press, 2004.