PRIORITY BASED BUFFERING OVER MULTIPLE LOSSY LINKS USING TCP AWARE LINK LAYER BUFFERING

THESIS

Kevin J. Savidge, Captain, USAF

AFIT/GCE/ENG/08-10

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the U.S. Government.

AFIT/GCE/ENG/08-10

PRIORITY BASED BUFFERING OVER MULTIPLE LOSSY LINKS USING TCP AWARE LINK LAYER BUFFERING

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science in Computer Engineering

Kevin J. Savidge, BSEE

Captain, USAF

March 2008

AFIT/GCE/ENG/08-10

PRIORITY BASED BUFFERING OVER MULTIPLE LOSSY LINKS USING TCP AWARE LINK LAYER BUFFERING

Kevin J. Savidge, BSEE

Captain, USAF

Approved:

_____/signed/__________________________ ________
Maj Scott R. Graham, PhD (Chairman) Date

_____/signed/__________________________ ________
Dr. Kenneth M. Hopkinson (Member) Date

_____/signed/__________________________ ________
Capt Todd R. Andel, PhD (Member) Date

Acknowledgments

I'd like to express my sincere appreciation to my faculty advisor Maj Scott Graham for continuing to be my thesis advisor, even after being reassigned. He dedicated his limited free time to mentoring and guiding me on my journey to complete this research. For this I am truly indebted to him. I want to also acknowledge the time and energy my family and friends spent listening to me discuss the difficulties I endured. I want to thank them for providing encouragement and for their endless support in my endeavors.

Kevin James Savidge

Table of Contents

Page

Acknowledgments ... iv

Table of Contents ... v

List of Figures ... viii

List of Tables ... ix

Abstract ... x

I. Introduction ... 1

Background ... 1

Problem Statement ... 5

Research Objectives ... 6

Investigative Questions ... 7

Methodology ... 7

Assumptions/Limitations ... 8

Implications ... 9

II. Literature Review ... 10

Chapter Overview ... 10

Terms Defined ... 10

Description ... 12

Relevant Research ... 14

Summary ... 22

III. Designing the Model ... 23

Chapter Overview ... 23

OPNET 14.0 ... 23

Model Implementation Overview..25

New Options..27

Small Network Anomalies 29

Packet Discarder Modification...34

Proxy Model Development..35

Memory Manager 37

Memory Scheme...38

Creating Priorities..41

Summary..42

IV. Methodology...43

Chapter Overview...43

Simulation..43

System Under Test 44

Factors 45

Parameters & Measurements..46

Network Modeling...49

Chapter Overview...51

Experimental Questions..51

Results 52

Investigative Questions Answered 59

Summary..62

V. Conclusions and Recommendations 63

Chapter Overview...63

Conclusions of Research 63

Significance of Research 65

Recommendations for Future Research....................65

Summary....................67

Bibliography 68

Vita....................70

List of Figures

Page

Figure 1. Propagation Loss across a Satellite with and without Buffering........................ 3

Figure 2. TCP Connection Establishment.. 12

Figure 3. Proxy Router Node Model.. 25

Figure 4. Clouds Used For Packet Loss.. 29

Figure 5. OPNET Project Using OSPF... 31

Figure 6. Revolving Buffer Example.. 33

Figure 7. Proxy Process Model.. 36

Figure 8. Final Scenario Design... 49

Figure 9. File Size Average Throughput Comparison with No Loss, Buffering Off 53

Figure 10. 25 Percent Loss, Priority Buffering Off, Large File size 54

Figure 11. 11 Priorities at 25 Percent Loss, Priority Buffering On, Large File Size....... 55

Figure 12. No Loss, a Large Files size, and Priority Buffering On, Logarithmic scale .. 56

Figure 13. No Loss, a Large Files size, and Priority Buffering Off 57

Figure 14. Average Throughput Buffering On an Off in No-loss Environment 58

List of Tables

Page

Table 1. Priorities by Port Number 41

Table 2. Factors 45

Table 3. Scenario Parameters 47

Table 4. Measurements 47

AFIT/GCE/ENG/08-10

Abstract

Wireless military information systems require high reliability, which is difficult to achieve in adverse conditions. To provide high reliability, one must overcome packet loss across multiple wireless hops. Buffering packets in a lossy environment is well explored; however, the ability to selectively buffer TCP traffic across multiple lossy links is a new area of research. This document seeks to explore the delivery of high priority traffic in a lossy environment and conclude that prioritized buffing can increase the probability that a high priority download will finish, where others will fail.

It is shown that buffering provides six times the throughput in a network with each link experiencing 25% loss. Prioritizing TCP packet flows provides a varied outcome, as it can not overcome the TCP mechanisms, when the packet loss recovery time is greater than the retransmission timeout event. However, the future work in chapter 6 may provide roadmap to gaining control authority of the challenged network.

BUFFFERING PRIORITORIZED TCP TRAFFIC IN A HIGH PACKET LOSS NETWORK

I. Introduction

Background

Military communication or information systems are essential for wartime decisions. When a military network is down, decisions to attack or defend are based solely on the most current information, which may be outdated. For operational success, it is often crucial that information systems be available, that data be accurate, and provide timely delivery. In a wired environment, with fixed network connections, these problems have largely been solved. In the highly dynamic environment of military networks, with connections changing regularly, and operating in volatile environments, several unique networking problems become exposed. Many different types of communication mediums are in use, from hardwired networks to laser communications. In addition, links may exist from ground to aircraft, ground to satellite, ground to ground, aircraft to aircraft, and aircraft to satellite. The majority of these systems rely on the Transmission Control Protocol (TCP) for reliable delivery of information.

The assumptions which went into the development of TCP did not include high packet loss networks [4, 6, 11]. The development of wireless networking, and satellite networking highlighted the shortfalls of TCP in such an environment. TCP provides reliable packet delivery from end-point to end-point in a dependable network and includes excellent throughput, fairness, and congestion control. TCP assumes packet loss results from congestion. In a wired network, packet loss generally occurs when too much information is sent to the receiver, overloading the incoming buffer or queue. The effect of overloading the queue is congestion. However, loss in a wireless network often results from more than simple congestion. Packets sent across a laser communications system may experience scintillation effects, or the loss of bits/packets due to atmospheric conditions such as dust or clouds which attenuate the power of the laser beam. The link may alternate between being up or down very fast, and with a high bandwidth link, multiple packets maybe lost in the interim. The packet loss is later sensed at the endpoints of the stream, through TCP mechanisms, which assume the loss is a result of congestion in the medium. In reality, the utilization of the link may be very low.

Some bit loss can be corrected in the physical layer through error correction schemes, but there are limitations. Once the number of bit errors exceeds the ability of forward error correction, the complete packet is lost. There are only two ways to recover lost packets. If forward error correction is insufficient, the sender must transmit a duplicate packet. This process is not a problem in a fixed, hardwired network with a small propagation delay. However, a mobile wireless network may entail transmission through multiple lossy links, resulting in low probability of successful packet delivery.

Retransmitting a lost packet all the way from the source can greatly increase the delay for that single packet. For example, consider Figure 1. Starting a time t_0, a packet leaves the source. Assume the packet is lost after being transmitted from the satellite. At time t_1, the lost packet should have arrived at the receiver. At time t_2 the acknowledgement didn't arrive at the sender causing the sender to resend at time t_3. The packet arrives at time t_4. If typical propagation time to a geosynchronous orbit is 125 milliseconds, then the retransmitted packet will arrive at time t_4 = 750 milliseconds (excluding additional transmission time and processing delays). If the satellite could buffer the packet, the packet would arrive at time t_2 = 500 milliseconds since the satellite discovered the loss and resent the packet. Thus, Figure 1 illustrates a primary benefit of buffering packets.

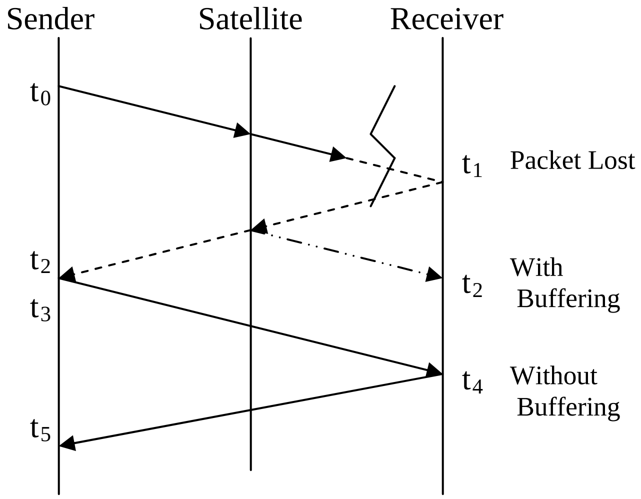

Figure 1. Propagation Loss across a Satellite with and without Buffering

Harmon's [1] and Reynolds' [3] research focused on reducing the effects of packet loss on TCP over multiple degraded links. Their work, discussed in detail in Chapter 2, is the basis for the research presented here. The basic approach is to store and forward TCP packets from router to router inside a challenged or high packet loss network.

In typical routers, a packet arrives and is forwarded on a path to the next router with the intent of closing the distance from receiver to destination. Routers are not responsible for reliable delivery of a packet. Routers generally "fire and forget" packets, leaving packet tracking to the end-points, typically using TCP. In a wired network, this technique alleviates the router from the significant processing overhead required to track packets and connections. In a wireless environment, more packets are lost as a result of link failures, resulting in delay and retransmission from the source. Reynolds and Harmon proposed buffering network packets on the routing devices, placing the burden of resending lost packets on the routing devices, and enjoying the benefits of lower delay and a reduction of retransmission load on the network.

As an analogy, consider a packet as a package to be delivered. A package sent via the US postal service is not necessarily tracked. The sender's package is placed in the mail and notification of delivery, if it happens at all, occurs when the receiver gets the package. If the package is lost, then the sender may send another package. Alternatively with tracking, each delivery point along the way accepts and takes responsibility for the package. The analogy breaks down in that the postal service can not duplicate a lost package as may be done in a network, but they do know the last place it was scanned, and have a better chance of finding it. In this latter case, the package is handled reliably between intermediate points. The buffering technique of Harmon works in the same manner, keeping track of packets from point to point between wireless links.

Harmon's data showed that his technique decreases packet retransmissions from the source and decreases the download time of a single 20 MB file transfer. The work also demonstrated that in high loss networks, such as with links with a 40 percent failure rate, TCP connections can still be maintained. Without buffering, the TCP connection fails. The proxy router models improved throughput and packet delay, as two benefits were accomplished. First, packets that were lost could be delivered closer to the point of failure. Second, TCP congestion control mechanisms, such as triple duplicate acknowledgements, were not employed. Timeouts were reduced as intermediate (proxy) routers could resend the packets. This protocol showed a reduction in transfer times and an increase in utilization, which is the motivation for continued work in this area.

Harmon also showed the effects of packet loss over a varied failure interval determined by a probability of failure. The results provide insight into the timescales of TCP's mechanisms, discussed more in Chapter 2.

Problem Statement

Given a challenged network, such as a laser communication system suffering scintillation effects, can strategically buffering prioritized flows increase the chances completing a download? Increasing the ability to transfer information in a wireless or challenged network is an area the military urgently needs. The need to be interconnected increases exponentially, as technology increases in the military. To stay interconnected in a mobile environment, wireless technology is required. Wireless technologies suffer from factors not prevalent in a wired environment. This research is focused on solving the problem of lost data on interacting networks of wireless links. The goal is to investigate the benefits of priority buffering, and understand the implications of buffering in a larger interacting network.

Prioritization ought to improve the quality of service for critical information. Although this research only explores TCP transfers, prioritizing routing information as critical may increase the capabilities of a mobile wireless network. Prioritizing and buffering can decrease the power needed on a sensor network by sending data once and only once, and only buffering high priority data when necessary, such as when buffer space is available and links are degraded. Prioritization provides a type of service level agreement.

This work extends the work of Harmon [1] and Reynolds [3] into a larger, interacting network. The improvement is modeling a proxy router with the ability to prioritize traffic flows based on a given priority. Traffic generated using TCP is given equal treatment through a routing device, since routing devices do not route at the transport layer. In Harmon's model, TCP packets are stored on the proxy routers until the next proxy router or destination acknowledges the receipt of this packet.

With this problem answered, this thesis concentrates on prioritizing individual TCP flows, and analyzing the results of delegating buffer space based on a given priority to a flow. Packets arriving to a proxy router will be prioritized and will attain buffer space based on the priority of the flow, availability of space, and time of arrival. The research is to demonstrate that in a degraded medium, a prioritized TCP flow given sufficient buffer space will be delivered more timely, and enjoy more throughput than a TCP flow with lower priority.

Research Objectives

The focus of this research is to model a medium to large network with multiple lossy links. The network will have many client-server pairs sending various TCP data transfers with varying levels of precedence. The results obtained from such a network should provide insight into the value of augmenting a router with additional memory and processing power in order to obtain a desired outcome in lossy networks. The results show whether the methods discussed provide a feasible solution to a challenged environment. If prioritization provides an opportunity to increase the reliability and timeliness of a network, then the outcome of this work may lead to an implementation in the real world. On the contrary, if prioritization fails to demonstrate or provide no further value to networking, then the reward is providing the knowledge to those in the industry to avoid the techniques presented.

Investigative Questions

There are many aspects of the research that are worth investigation. Simulation time and memory will always set the limits of answers. For this research, the most important question is whether or not strategically assigning buffer space to a prioritized stream of data will allow the higher priority information to be successfully delivered in the event the link is extremely poor. As the outages are increased and packets are dropped, can buffering continue to keep a TCP connection alive and finish a high priority download?

Methodology

Building upon predecessors' models, this research investigates the value of strategic buffering. Using TCP aware link level buffering to increase the network efficiency, we apportion limited memory based on given priorities. The model is simulated on the OPNET 14.0 network simulation tool.

Assumptions/Limitations

This research assumes the use of the ubiquitous Transmission Control Protocol. While the congestion control mechanisms tend to thwart efforts to employ intermediate buffering, the benefits to enabling a buffering mechanism which can utilize TCP are diverse. It is desired that the research not affect the ability, fairness and congestion mechanisms of TCP outside the challenged portions of the network. TCP outside the controlled network must perform as advertised. The changes to the routers in the controlled network must be invisible to the outside network, other than providing the perception of a more reliable network, albeit with some delay.

The models provided created by Harmon [1] use the TCP sequence numbers to provide accountability between routers. Hence, the routers will focus on the transport layer as the transport layer is responsible for the reliability of packet delivery. Although there are techniques in this research that could be applied to lower levels, TCP, a transport layer protocol, is the focus of this research.

Another key point is to apply the buffering in a multiple lossy links or challenged environment with multiple routing devices. Most of the protocols in Chapter 2, such as Snoop TCP, discuss an implementation from the last hardwired link to an endpoint over a single lossy link. For this implementation, there are multiple hops with multiple lossy links.

Implications

Hiding network loss in the link layer could impact other functions in the network. Buffering and prioritization must therefore be approached with caution so as not to affect the fairness or congestion mechanisms. A problem such as channel capture, where one device dominates the medium, could reduce throughput for all other devices, reducing the greater good for the network. Link layer buffering may also increase flooding of the network, wastefully sending the same packets over and over.

Some internet security protocols encapsulate the TCP layer, thereby encrypting the TCP information. The TCP aware link layer buffing protocol discussed here will not work with security protocols that encrypt TCP information. It is presumed that this protocol can be employed on IP protocols, with some additional development beyond the scope of this research.

II. Literature Review

Chapter Overview

The purpose of this chapter is to discuss relevant research. The first part of this chapter discusses the research and techniques employed by the networking community which led to the model built by Harmon. The second part of this chapter discusses in detail the research done at AFIT by Reynolds [3] and Harmon [1]. Their research is the baseline for the work presented. This chapter should provide the reader with better understanding of the present model.

Terms Defined

Many of the research papers discussed in this chapter define the same ideas with different terms. For the reader's benefit the related terms are defined here for quick reference.

Acknowledgements, (ACK, ACKs) are TCP generated responses for the successful reception of a packet of information. ACKs arriving at the sender provide TCP the feedback loop to ensure reliability in the transport layer.

AckPings and *Persist Requests (PR)* are terms to describe the proxy router's special packets for discovering lost ACKs. In a degraded link, packet loss includes the acknowledgements, so a proxy router will ask its neighboring proxy if it sent an IACK already. If it did, it will respond with an *AckPingResponse* or *Persist Response*.

Challenged Environment is a network with multiple hops each having degraded links connecting them. Challenged environments may be mobile networks, wireless fixed networks, or satellite networks.

Challenged Links, Degraded Links, Lossy Links, and *Link-Winking* all describe packet loss on a link. There are discussions of the ability of links to fail and recover in nanoseconds. Link recovery may be a predetermined interval of time which some argue invalidates the result of prior research. The overall intent, whether or not a link can recover in nanoseconds, is the loss of packets. Packet loss due to bit errors, forward error correction (FEC) failure, or scintillation effects is the focus of this research.

Flows are single TCP connections from client to server over a single port. Flows consist of the connections handshaking, acknowledgement and data packets.

A *hop* or *node* defines a smart network device such as a switch or router proxy router that accepts a complete packet, determines a path or output for the packet and forwards the packet.

Intermediate Acknowledgements or *Local Acknowledgements* (IACK, LACKS) are similar to ACKS but they acknowledge a packet has been accepted by an intermediary device.

Link Winkers or *Packet Discarders* destroy packets that are transmitted across the link when the link is considered down. Configuring IP clouds packet discard ratio will produce similar results, but the IP cloud does not allow a start and finish time.

Proxy Routers, intermediate nodes, and *strategic buffers* all describe a router with the ability to store packets.

A collection of node pair names for traffic generation and collection are scattered throughout. The terms *server*, *sender, source* all refer to the traffic generator and the terms *client*, *receiver*, *destination* all refer to the traffic collector.

Description

Transmission Control Protocol

RFC 675 “Specifications of Internet TCP” is dated as December 1974. Although networking has changed significantly since that date, the basic operation of TCP remains unchanged. The Transmission Control Protocol provides a connection oriented, reliable, byte stream service to the application layer. The term connection oriented means the two applications using TCP, such as a client-server, must establish a TCP connection with each other before they can exchange data [1]. After the three way handshake shown in Figure 2, the server sends data and the client responds with acknowledgements for the data.

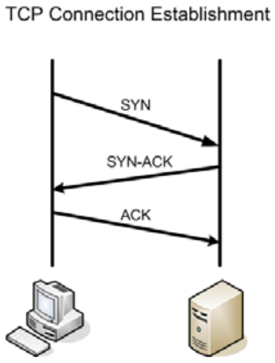

Figure 2. TCP Connection Establishment

This is the vanilla TCP framework, but additional mechanisms were added to TCP as more networks were joined to overcome channel capture, fairness, and congestion. This led to the many ‘flavors’ of TCP such as Reno, Vegas, Tahoe, etc.

In a wired environment, packet loss due to physical damage or power failures is <<1 %. Therefore, lost packets are indicators of congestion in a wired environment [1]. Congestion is the arrival of packets at a device at a faster rate than the device can process them. Generally speaking, routing devices typically have a queue for incoming packets. The size of the queue and the processing abilities of the router determine how fast packets can arrive at the device. If the rate of packet arrivals exceeds the service rate, the queue fills up. All packets arriving to a full queue are lost.

There are two indications of congestion in a network, a timeout event and duplicate acknowledgements [1]. TCP has a Retransmission Timeout (RTO) event which is based on the Round Trip Time (RTT) and a smoothing algorithm from end to end. At the senders end, if an acknowledgement takes longer than the RTO, a duplicate packet is sent and the RTO is reset and multiplied by an exponential backoff. The exponential backoff provides additional time for the receiver to acknowledge the receipt of the duplicate packet.

At the receivers end, if a packet is lost or out of order, the receiver sends a duplicate acknowledgement. This alerts the sender that either a packet is lost or out of order. In the latter case, out of order packets are immediately acknowledged by the receiver since the missing segment may be lost and the transport layer can not deliver packets to the application layer unless the packets are in order. Packets aren’t guaranteed to cross the network in order so the sender waits until it receives three duplicate acknowledgements for the same segment before resending a lost packet. This avoids sending redundant packets when the packet arrived out of order.

There are numerous papers [8, 11, 12, 10] on reliable transport protocols in degraded operating environments. A degraded, lossy, or challenged network discussed in this thesis is one that suffers intermittent packet loss due to the environment or mobility. These challenged networks are using protocols designed for wired links in the early 80's. These protocols have been tuned for traditional networks made up of wired links and stationary hosts [4]. Typical probability of bit error in a hardwired system is 10^{-8}-10^{-12} whereas losses in a wireless system are 1 in 10^{-3}-10^{-6} [8]. The Transmission Control Protocol was designed to treat intermittent failures as network congestion. For a router to truly lose a packet in a wired network, the router's incoming buffer or queue must be filled and unable to handle the incoming request. When buffer overflow occurs, TCP was designed to back-off or return to a slow-start state to avoid adding additional congestion. This mechanism creates numerous problems when employed on a wireless network. The following paragraphs will discuss the related work on challenge networks.

Relevant Research

Improving TCP Performance

There are two fundamental approaches to improving TCP performance in a challenged network. The first approach hides non-congestion related losses from the TCP sender, and therefore requires no endpoint modifications. The second approach is to make the sender aware of the non congestion related losses [11]. This approach requires the endpoint to be modified to accept explicit notifications. Again, the objective of this research is to operate without changing TCP at the endpoints, so the first approach is modeled. Work in [11] additionally classifies each scheme can into three sub groups. These are: an end-to-end philosophy, split-connection proposals and link-layer proposals. The end-to-end philosophy attempts to make the TCP sender handle losses through explicit loss notifications, or through selective acknowledgements (SACKs). The split connection philosophy hides the losses by creating multiple individual connections. A base station at the edge of the challenged network creates a TCP connection with the sender, and in-turn creates another connection across the challenged link. The last philosophy [11] discusses is a link layer solution. The link layer resends lost segments locally using acknowledgements, and may use forward error correction (FEC) to recover packet losses. This philosophy hides losses from the TCP, but may not be able to hide all losses. Work in [11] evaluates many of these techniques in combinations to gain insight on the benefits of each and is highly recommended reading for further understanding. The model presented in this thesis hides the non-congestion losses from TCP. However it doesn't fall directly into the three sub groups discussed by [11]. The model uses the transport layer header to control acknowledgements and data resends at the link-layer. This categorizes it as a TCP aware link-layer recovery mechanism. The technique is similar to link-layer recovery, but link-layer recovery discussed in [11] doesn't distinguish TCP flows and doesn't provide in-order delivery of the packets.

All the techniques discussed are labeled as Protocol Enhancing Proxies (PEPs). PEPs as discussed in RFC 3135 are highly discouraged because they may interfere with the original intentions of the protocol, such as fairness of the shared link. However, in limited circumstances, such as discussed in Chapter 1, military networks may be an appropriate environment for PEPs because of the unique circumstances. Deployed military networks do not operate in a service provider capacity. Military networks are generally located on the edge of large networks and are mostly sovereign.

DTN Protocol

Closely related research at Berkeley by Fall [7] discuses Delay Tolerant Networking (DTN) which specifically avoids PEPs. The concept is to create a gateway responsible for a region. Traffic passing through the region is treated in a manner similar to Simple Mail Transfer Protocol (SMTP). In SMTP, a message is sent from mail server to mail server until it reaches the server responsible for the recipient. The recipient mail server takes responsibility for the message and the sender is relieved of further duties. The DTN is a similar concept that uses application layer processes to accept responsibility for a region or from hop-to-hop. DTNs guarantee delivery and reduce traffic as retransmissions over multiple hops are eliminated. DTN is a concept which specifically avoided PEPs as discussed in RFC 3135. This concept works when emphasis is on the reliable delivery, and the DTN concept is very close to the research presented here. However, as mentioned in the limitations, this model can not change TCP. DTNs also present the same issues when sending large files across our packet dropping medium. As DTN is an application layer, the underlying transport protocol will still be TCP. The benefit of DTN is similar to the research done by Harmon [1]. The difference is that the proxy routers provide a packet level guarantee of delivery from hop-to-hop, instead of accepting a complete file. DTN also creates a limitation as the complete file would have to be received, to ensure it wasn't corrupted before it is resent. This would increase the size of the buffer or memory needed. DTN works regionally so if there are numerous challenged links between regions, the problem still exists.

Snoop Protocol

The Snoop Protocol [4] paper is closely related to the work of Harmon. Snoop works by caching or buffing packets at a fixed host and has the capability to resend these packets if the mobile host doesn't receive them. Snoop accomplishes this by altering the fixed host protocol to capture acknowledgements from the mobile host. Capturing acknowledgements allows the fixed host to resend the packets before the sender times out. Additionally if a sender receives more than three acknowledgements for the same packet, TCP will reduce its output in half. Snoop alleviates this congestion control mechanism, allowing TCP to continue sending at the current rate. TCP will also time out if no acknowledgements are received. Snoop can handle a smaller time out window than TCP and will retransmit to the mobile host multiple times before the sender times out. This increases the efficiency of the network, since a time out event causes TCP to return to a slow-start state. However, Snoop's design, as discussed in [4], works from the last fixed station to the mobile host. Snoop is a last mile scenario which leaves a big hole in our wireless networks which may stream through a multitude of challenged links.

Indirect TCP

Indirect TCP (iTCP) [17] is another alternative closely related to this work. iTCP creates a separate TCP session between the wired link and each individual wireless hop.

Since each individual connection over the wireless hops uses TCP, they can fail and cause the original sender to stall. Each wireless hop creates a TCP connection. Inherent issues with this protocol include acknowledgements being sent back to the sender before the data packets actually reach the destination. This violates the semantics of the TCP acknowledgements [11].

Split TCP

Like iTCP, Split TCP [20] creates individual TCP connections from router to router. It buffers packets on proxy routers and forwards them at the rate of acknowledgements from the recipient of the packets. Packets are cleared from the buffer when an ACK is received from the destination. From what is proposed, it appears that the end-point TCP is not changed. However, the paper discussed the source sending data at the rate of receiving LACKs. To understand and act on the LACK, one of two methods must be employed. One scenario is the LACKs are no different then ACKs and the source acts upon them moving the send window ahead. In this case the semantics discuss in iTCP are violated, a packet is ACKed before it reached the destination. The second scenario is the LACKs are differentiable from ACKs, which means the end point TCP must be changed to accommodate the LACKs. End-point changes are outside the constraints placed on this research.

Split TCP is the first model that discusses a TCP solution over multiple challenged links. Split TCP is the closest protocol to the model derived in this research. Harmon’s model created a reliable delivery from router to router but does not require transport layer connections between devices like Split TCP requires.

Reynolds Model

Reynolds' [3] research focused on the transport layer TCP buffering. In his model, intermediate nodes or routers had buffering capabilities. The routers are located close to the challenged or 'winking links.' Packets sent across the network are buffered in the router and the router acknowledges the packets by sending an intermediate acknowledgement (IACK) to the TCP source. Packets lost on the challenged link are resent, from the first unacknowledged packet to the last buffered. When the packets are received by another buffering router an IACK is sent back and the acknowledge packets are removed from the buffer. When the packets are received at the destination, the TCP acknowledgement (ACK) is sent back clearing all the buffers and advances the TCP window of the source.

An IACK is similar to a TCP ACK in that it tells the TCP connection that the packet has been received, but it doesn't allow the TCP window to move. Instead an IACK informs the TCP source that the packet is buffered and not to resend the packet. An IACK can manipulate the window size making TCP send additional packets to be buffered. Reynolds model accomplishes three goals. First, it decreases the number of duplicate packets, freeing the link to be used for transmission of new packets. Second, it moves the packet closer to the point of failure, so the propagation, transmission, queuing delays, and processing delays incur from sending a packet from the source are eliminated. Third, his buffering scheme reduces the back-offs incurred by TCP when packet loss is confused with packet congestion, thereby increasing the throughput of the flow.

The principle constraints of this thesis are broken when TCP is modified. Reynolds' work included modifying TCP protocol to understand the IACK. His version of TCP also allowed the IACK to increase the window size. These methods require all end points to modify the TCP. Harmon [1] incorporates the buffering ideas along with the IACK, but only the routers understand the IACK not the endpoints. The techniques of hiding the network losses by buffering and providing the reliable handoff of packets between routers is the basis for Harmon's model.

Harmon's Model

Harmon's work [1] is a TCP aware link layer protocol, specifically designed to increase TCP performance and enhance the reliability and end-to-end delay without disturbing the protocols in the external systems. The capabilities of link proxies are to store and forward packets and retransmit the packets locally if lost. Proxy routers must have fast memory and be capable of processing acknowledgements and retransmission requests.

TCP expects the packets to be acknowledged within one round trip time plus an offset. If the ACK is late, the source will automatically send a duplicate packet. This effect is a retransmission timeout (RTO). The second mechanism that causes a resend is the triple duplicate acknowledgment (TDA). In the link proxy router concept, the proxy routers have two mechanisms to counter the RTOs and TDAs. The first is an event timer which works similarly to the RTO. The event timer is an RTO for the intermediate link between to proxy routers. This RTT is much smaller than the TCP RTO; therefore the proxy router event timer is smaller and may send the packet multiple times before the

TCP RTO event. These event timers are waiting for an IACK from the adjacent router, if one is not received within the event, the packet is resent. The second mechanism is the ability to act on TCP ACKs. If a proxy receives a duplicate ACK for a packet buffered on the router, it will resend the packet immediately. Since the proxy routers are closer to the destination than the source, a triple duplicate acknowledgement is less likely.

Acting upon the duplicate acknowledgements and local resends creates a three fold benefit as explained in Reynolds' [3] model. First, local resends reduce the traffic across the good links by resending the packet closest to the point of failure. Second, local resend allows the mediums not affected to use the bandwidth to send new packets. Third, it allows TCP to maintain it present throughput by avoiding a third duplicate acknowledgement.

TCP ACKs are sent for packets that reached the destination successfully. The acknowledgement for that packet must be handled by each proxy to clear the buffer of that packet, making room for additional packets. The proxy routers set timers for the ACKs, and if the timers expire they resend the buffered packets, to avoid a RTO at the source. Harmon's model maintains state information for each flow passing through the proxy router. A flow is defined as a TCP source destination pair distinguished by a 4-tuple (Source IP address & port, Destination IP address & port). In a large network, maintaining a list of every flow, packet, and sequence number is extremely taxing on a processor. In a fixed network millions of flows may be happening at the same time. This research assumes that proxy routers will not be used in fixed environment with millions of customers.

Maintaining the TCP flow connection record is quite intricate. Each connection record must contain the 4-tuple, sequence numbers, sequence number gaps, ACKs, IACKs, memory allocate, and memory used. The structures necessary to track the flows adds processing time and memory requirements. The limitation of additional memory creates constraints on the simulation software discuss in more detail in Chapter 3.

Summary

This chapter covered the background information necessary to understand the choices for the model. The next chapter will discuss the optimizations made to Harmon’s model, and the methodology of creating a priority based buffering system.

III. Designing the Model

Chapter Overview

The purpose of this chapter is to discuss pertinent information on the model. How the model was employed in the network is discussed in Chapter 4. There are many intricate details provided about the OPNET model developed by Harmon discussed in Chapter 2 and in greater detail in [1]. Between the time Harmon finished his research and this research began, Matt Weeks [23] took on the project to refactor the code. Most of the code was externalized from OPNET and some additional features are discussed below, which are relevant to the reader as they change the operation and process flow.

OPNET 14.0

OPNET simulates packet transmission through data structures which hold information such as the headers, amount of data in the packet, and time stamps. It is important to understand that the connection record is only a pointer to the memory structure. Experiments proceed decoupled from real time. To keep the experiments simple and predictable, processing times such as memory access were not incorporated into the model.

The wall clock time or real time it takes to simulate a discrete event in OPNET can vary greatly. Since the software is simulating multiple events happening at the same time, the simulation time does not increment until all events scheduled for that time are completed. Smaller network simulations might simulate weeks of network traffic in 20 minutes of wall clock time. More intricate models might require hours of real time to simulate seconds of network traffic. One advantage of using simulation software is the ability to send only pointers, and not the actual data. Mathematical models can statistically derive the time it takes for a single packet to traverse the network with out sending it. When the number of events needed to simulate the model outweighs the advantage of not sending the actual packets, the wall clock time increases.

Additional factors such as model design can cause the simulator to spend much of its time handling interrupt timers. In the packet discarder model discussed later, the algorithm randomly determines if the link is up or down during an interval. At large intervals of 100 milliseconds, this effect does not impose much of a penalty, around 1000 determinations per second of simulation time. As the intervals are shortened to 1 microsecond the penalty grows very large. For every second of simulation time, 1 million determinations are required. As the number of discarders in the network grows, the simulation software carries a large processing penalty. The packet discarder is a simple example that can be calculated before design, but some design implementations are not obvious until after simulation. The processing penalties are typically due to the inherent design of the simulation software, usually not known to the designer until after a problem occurs.

As found during some simulations, physical memory constrains the ability to model large networks in a discrete environment. One network with 30 client-server pairs and 15 routers running the buffering protocol consistently failed as a result of exceeding available memory allowed by the operating system. This is a valid constraint that needs more investigation beyond the scope of this thesis. Optimizations were added to the link layer buffering design to accommodate the issues presented, but these limitations still constrain the size of the network.

Model Implementation Overview

The proxy router was created from an OPNET ethernet4_slip8_gateway model. The ethernet4_slip8_gateway is a generic router model supplied by the simulation software with 8 point-to-point protocol connections and 4 Ethernet connections.

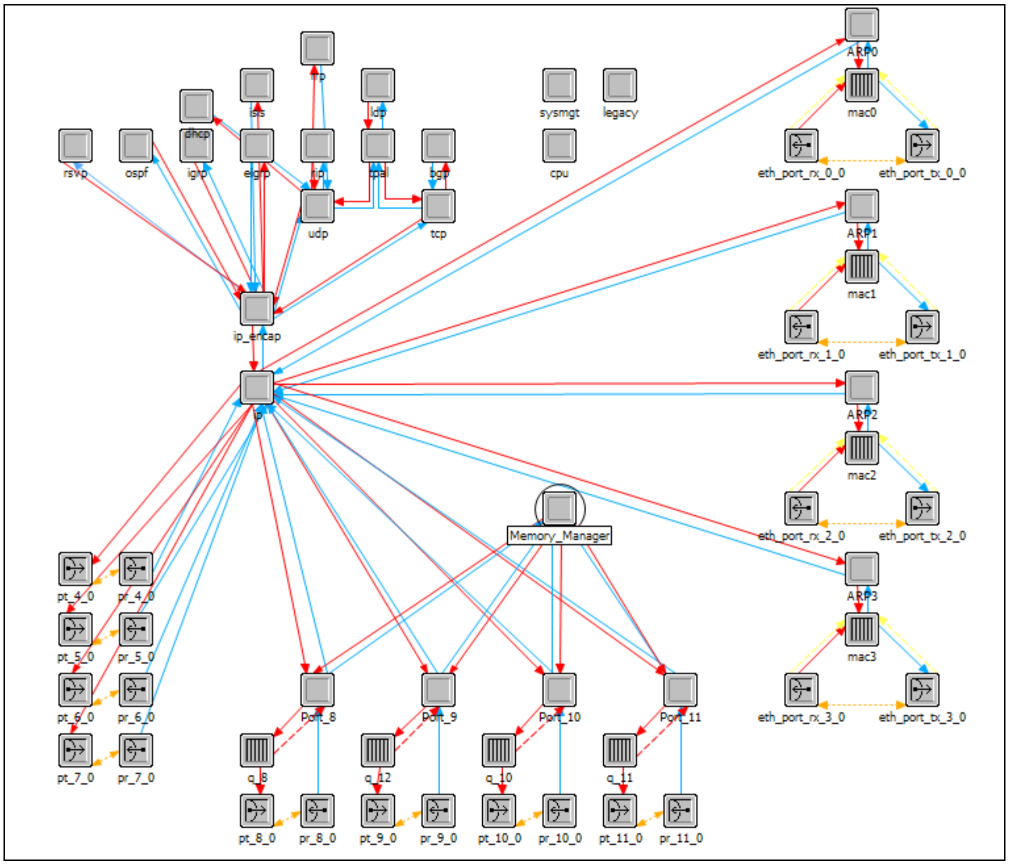

Figure 3. Proxy Router Node Model

Proxy routers use the generic router with the addition of the four proxy process models, a memory manager and four outgoing queues shown in Figure 3. The proxy process model intercepts all incoming packets and acts only on two protocols, TCP and Proxy. All other protocols are simply forwarded after being inspected.

Newly arriving TCP data packets are forwarded to the IP layer for processing. All buffering is accomplished on the outgoing link. When a new packet is received by the proxy process model from the IP layer, a record identified by the 4-tuple is created. The proxy process model then sends a request packet to the memory pool manager requesting buffer space for the packet. The memory pool manager processes the request, and sends a response packet back with the amount of memory allocated. The proxy process model then records the amount of memory allocated by the memory manager and the amount this segment needs. The packet is forwarded and an IACK is sent to the preceding node.

The model checks for a connection record before creating one. If a connection record for a flow already exists, and memory is available, then incoming packets are stored without contacting the memory manager. However, the memory manager keeps a separate record of the memory allocated and memory used for each link. An external function updates the memory used in the manager’s table. The drawback to this design is trying to keep parallel list structures. If the memory manager’s record is not an accurate reflection of the proxy’s record then buffer space is wasted. If the records do not mirror each other, then unexpected results such as null pointer events occur. The choice to have an external function update the memory manager alleviated additional notification packets from being sent from the proxy process model to the manager for every stored and deleted packet. This reduced the complexity of the manager's process model and reduced simulation time.

Harmon's [1] memory pool manager process model gave every request 10 packets of buffer space if memory space was available. This process reduced the number of request packets sent to the manager. When the 10 allocated packets are filled the proxy process model sends another request packet. If memory is available then the manager responds with another 10 packet allocation. This methodology can be an advantage or disadvantage. Since the memory is limited, the strategic buffering plan needs to be greedy with the memory it allocates. Further investigation showed that the flows were not using the 10 allocated buffer spaces, and the memory manager was quickly running out of free memory. The new default memory allocation standard is 2 packets, matching the number of outstanding packets in TCP slow start. If additional space is needed, then 2 more (2920 bytes) are allocated. The shortcoming is the additional time required to send and process requests for more space when the priority algorithm is employed.

Acknowledgements arriving from the port receiver are intercepted by the proxy process model. The 4-tuple is examined along with the sequence number, ACK number and the date length. This information is used to look up buffered packets. If the acknowledgement is for more than one packet, then all packets acknowledged are removed from connection record.

New Options

Weeks added a new queue to the outgoing stream along with a new selective intermediate ACK (SIACK). As discussed in Chapter 2, out-of-order packets cause the receiver to immediately send a duplicate acknowledgement for the missing segment. For each additional out-of-order packet, the receiver sends a duplicate acknowledgement for the missing packet. Three out of order packets causes a TDA. To avoid a TDA, when a proxy detects a missing packet, it immediately sends the preceding proxy a SIACK. The preceding proxy can then reorder the outgoing packets in the queue to send the missing packet without delay. The receiving proxy places the missing segment in the queue in sorted order allowing the flow to be transmitted in order.

The second addition is an option for asymmetric links. In a real network, the path the data packets traverse is not necessarily the path on which the acknowledgements return. Asymmetry creates an issue if the proxy routers do not receive the ACK for the packets buffered. Harmon's model uses the ACKs to clear the buffer of the very last proxy closest to the receiver. If the proxy does not receive the ACK, it will continue to resend the packets. To alleviate this situation, timers and a 'buffer downstream' option was added. There is a timer for each flow and if no ACKs or data packets are detected it is assumed the connection timed out. The buffer downstream option allows the routers at the edges of the network to adapt. The buffer downstream option turns off the proxy protocol going to the outside network or to any device. The algorithm detects if the next hop isn't a proxy by setting a timer and waiting for IACKs. If after a reasonable amount of time, IACKs are not received, or if an ACK is received before an IACK then the buffer downstream option is turned off. This achieves two benefits. First it stops sending duplicate packets from the challenged network to an outside network which can not IACK. The transmission and propagation delay to the receiver may be long enough for the last proxy to flood the network with duplicate packets. The second benefit is to save buffer space, if the next hop can not IACK, buffering at the edge is moot. The buffer at the end node would become extremely large. If properly placed in the network, the last proxy should be connected to a fixed environment. In this research, the last proxy model will send intermediate acknowledgements to the previous proxy in the challenged network. This process will clear the previous buffer. Again, this option achieves the objective of not affecting the outside network, with transmission of unnecessary duplicate packets.

Small Network Anomalies

After exercising the OPNET model in a network with two paths from source to destination, multiple anomalies were observed. The models were placed in a simple example shown in Figure 4.

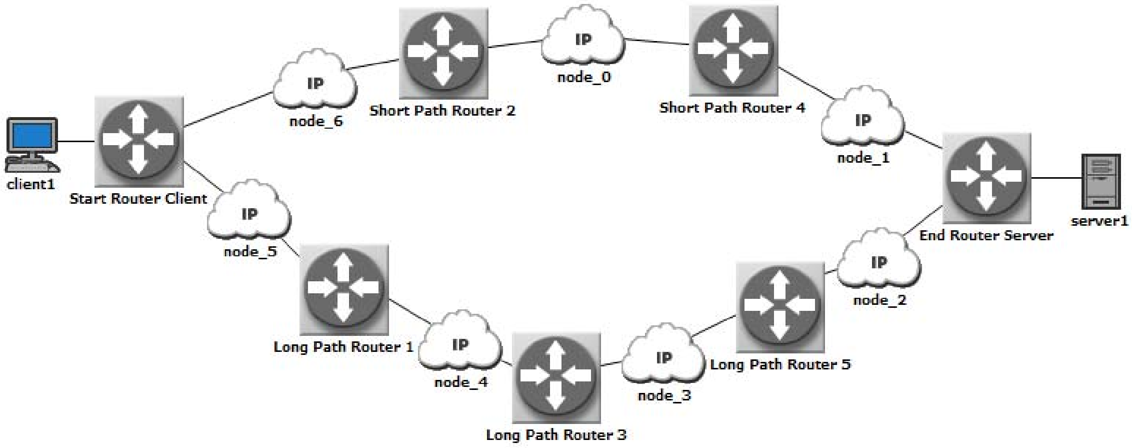

Figure 4. Clouds Used For Packet Loss

Open Shortest Path First (OSPF) and Routing Information Protocol (RIP) are employed to build the routing tables. Ip_clouds, an OPNET provided model, were implemented to simulate the propagation delay and packet discarding ratio. An ip_cloud is a router with the ability to set, among other parameters, a fixed amount of packet latency, and an average discard ratio. Increasing packet discard ratio beyond 15% in the ip_clouds defeated routing table updates. Therefore without routes, TCP could not establish a connection. The basic ip_cloud for OPNET does not provide an option for setting the time at which it begins to drop packets. Hence, it is impossible to exercise higher drop ratio scenarios using the ip_cloud construct. Therefore, instead of the ip_cloud, a custom packet discarder is used. Configuring the packet discarder to begin dropping packets at 180 seconds allowed sufficient time for the routing tables to update.

A second experiment exercised the effect of breaking FTP transfer in midstream. The usefulness of this simple example is the ability to simulate the asymmetric options added by Weeks. Two paths were constructed with exactly the same bandwidth, except than an additional hop was added in the lower link, giving the top link a shorter path. During the simulation another anomaly presented itself which led to a small change in the OSPF settings. The model is presented in Figure 5. All links between routers are good links with no loss, except for the link between short path router 2 and short path router 4. The FTP transfer was disrupted by severing the top link between routers 2 and 4 about halfway through the FTP transfer to force a routing table update. Unfortunately, the OSPF timers were set to determine a ‘dead link’ after 55 seconds. The FTP transfer failed, long before the routing table updated. To fix this problem OSPF timers were reduced to determine a ‘dead link’ in 6 seconds. Setting the timers this low in a real network would generate excessive routing update packets (OSPF hello packets) between adjacent routers, but it provided a method to validate the asymmetric feature.

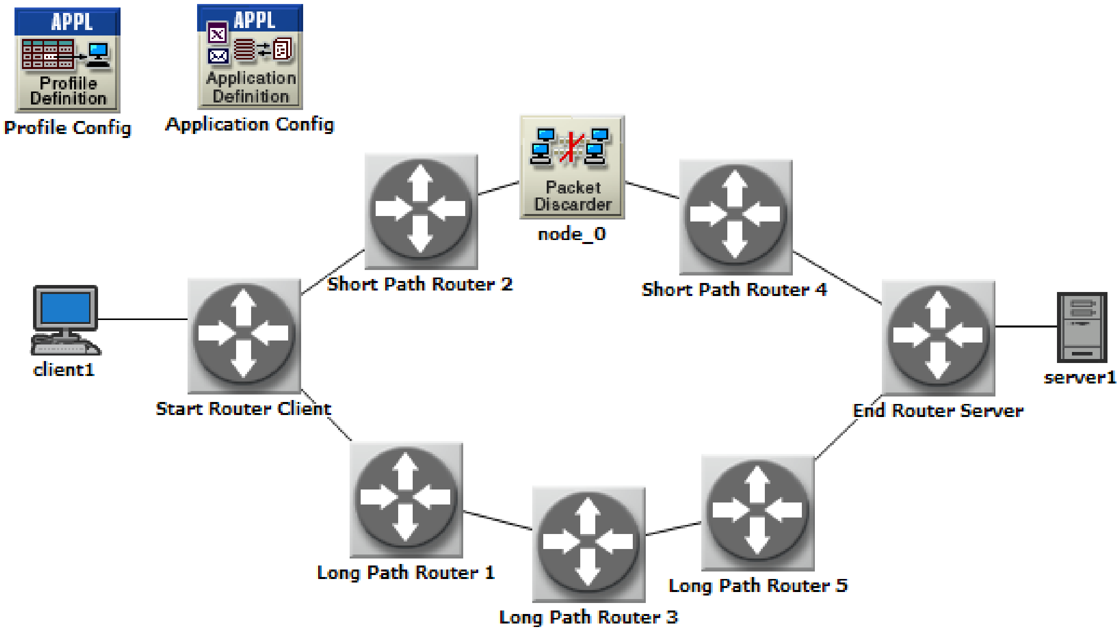

Figure 5. OPNET Project Using OSPF

After the top link is severed and the routing tables are updated, the first TCP action is a RTO on the server. The server sends the next unacknowledged packet to the client. The client already received this packet and sent the acknowledgement on the old link where it is still buffered on proxy short path router 2. This packet is rejected by the client and a new acknowledgement is sent, requesting new data with a sequence number two packets greater along the bottom path. Since this acknowledgement’s sequence number was higher then the all the packets the bottom path observed, it was ignored. The server received this acknowledgment, and adjusted the TCP window sending the next packet. Unfortunately, on the proxy routers, the RTO data packet is stored, along with the new packets, with a gap of two packets or 2920 bytes. This anomaly caused the extreme growth of packets on the buffer, as they need an IACK for the missing packets to clear the buffer. On a single run, the buffer maximum size was granted at 10,000 packets per proxy router. At the end of the simulation, 8000 packets were buffered on long path proxy routers. This crucial abnormality caused a major change in the model. If buffer space is limited then useless packet storage is counterproductive to the object of this research.

The receiver advertises the maximum window size to the sender when a connection is made, and adjustments are sent on subsequent acknowledgements dependent on the receiver's ability to process packets. In the OPNET model the maximum window size is set at 65536 bytes, or ~44 packets. This number is based on the 16 bits in the TCP header. The maximum window size determines the number of outstanding segments or the maximum number of packets in flight. Applying the windows scaling option defined in RFC 1323 allows more then 65536 bytes, but for simplicity in the OPNET coding, the maximum window size is fixed at 65536 bytes or 44 packets.

The 45th packet to arrive on the buffer informs the proxy that the lowest packet on the buffer is already acknowledged by the receiver. This proposal is guaranteed, otherwise the sender's window could not move.

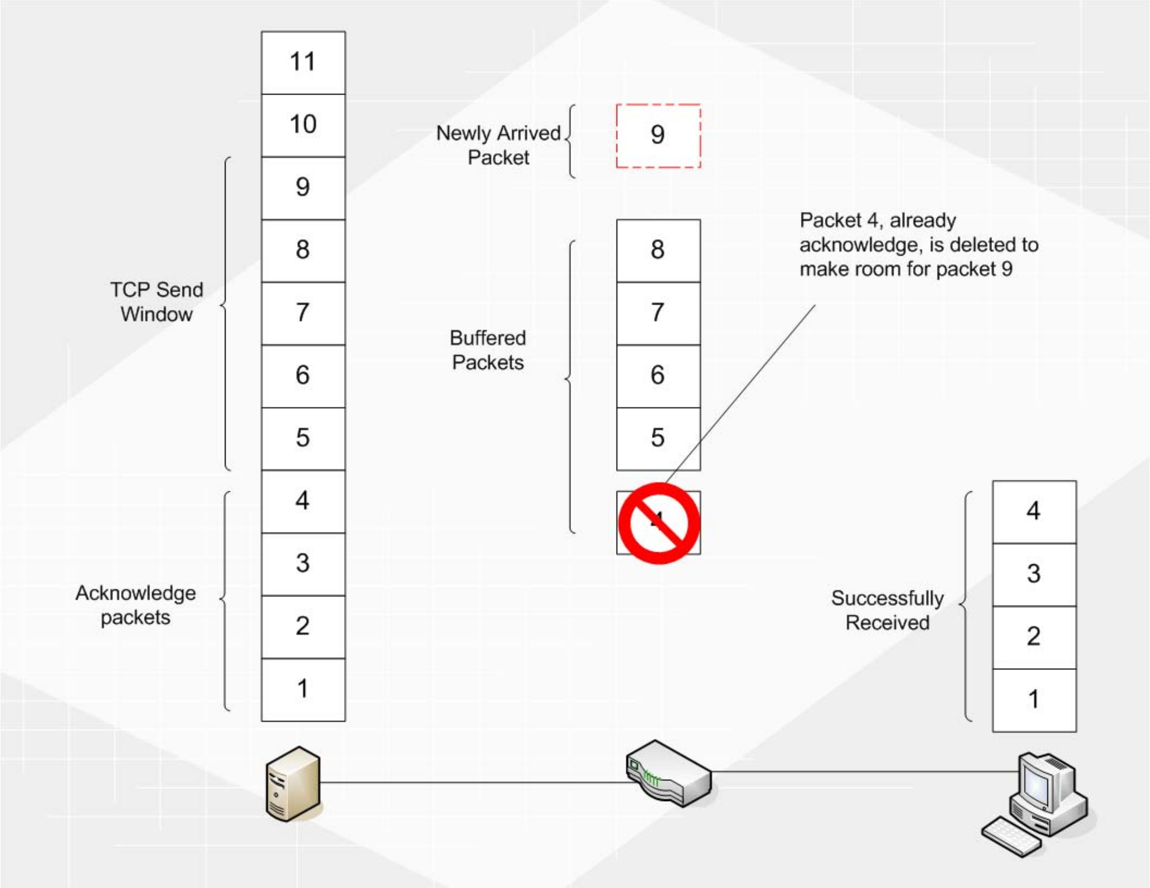

Figure 6. Revolving Buffer Example

For simplicity, assume that the max window size is five packets, reference Figure 6. The proxy's buffer then only allocates space for exactly 5 packets per TCP connection. The server is sending 11 packets to the client, with a proxy router device in between buffering the packets. When packet four arrives at the receiver, an acknowledgement for packet four is sent back to the sender. If this acknowledgement is not seen by the proxy, but still accepted by the sender, then the sender's TCP window moves forward and the next five packets are available for delivery. Since the window is controlled by TCP, the ninth packet delivery acknowledges the forth packet. The model now incorporates a revolving buffer where the 50th packet causes the first packet in the flow to be destroyed. 50 packets were chosen to provide protection in case the segments are not 1460 bytes in size. Using the sequence numbers, data length and max window size is truly needed to calculate the actual amount of buffer space per flow in bytes. Incorporating this in an OPNET model became too cumbersome, so for implementation 50 packets provided enough buffer for analysis. For the simulations, the TCP segments are normally 1460 bytes in data length except for the acknowledgements. Fragmented packets are not handled by the model because the proxy routers would need to reassemble fragmented packets before they could be resent. This is beyond the scope of this project due to complexity of the OPNET model.

Packet Discarder Modification

The packet discarder developed by Reynolds [3] and Harmon [1] provided a deterministic packet dropping link. Their packet discarder uses a set interval and a uniform distribution to determine if the link is up or down. Reynolds research needed a deterministic interval as he attempted to estimate the link up time. The goal was to strategically calculate when the link was up.

Harmon's model incorporated the same algorithm to study the effects of TCP congestion mechanisms over set intervals and probability of failures. Since this research is focusing on the buffer management scheme and the prioritization of flows, a change to the packet discarder was necessary. The interval needed to be an exponential distribution to provide a realistic time between failures, and a realistic length of failures. The exponential distribution models a continuous time between changes in state, which fits best when describing intermittent packet loss on the link. Again it is expressed that this degradation in the link is not equivalent to a link failure. The wireless links modeled may intermittently drop bits or packets depending on the interference, but the link itself is not broken.

To incorporate the change, the interval used by Harmon is used as the mean of the exponential distribution. The probability of failure is still a uniform distribution. When the packet discarder is initialized it is set to start at a specific time in the simulation. This allows time for the network to build the routing tables. The first exponential interval is chosen at the start time. The uniform distribution determines whether the packets will be dropped or forwarded during the interval. An interrupt is set, and the simulation will return to repeat the process when the interrupt occurs. This new technique provided an increase in efficiency of the model, since the number of interrupts is reduced.

Proxy Model Development

The process models are well described in [1]. Changes made to the process models were necessary to incorporate the recovery of memory when the flows were not using it, as well as the ability to recover from the loss of a proxy.

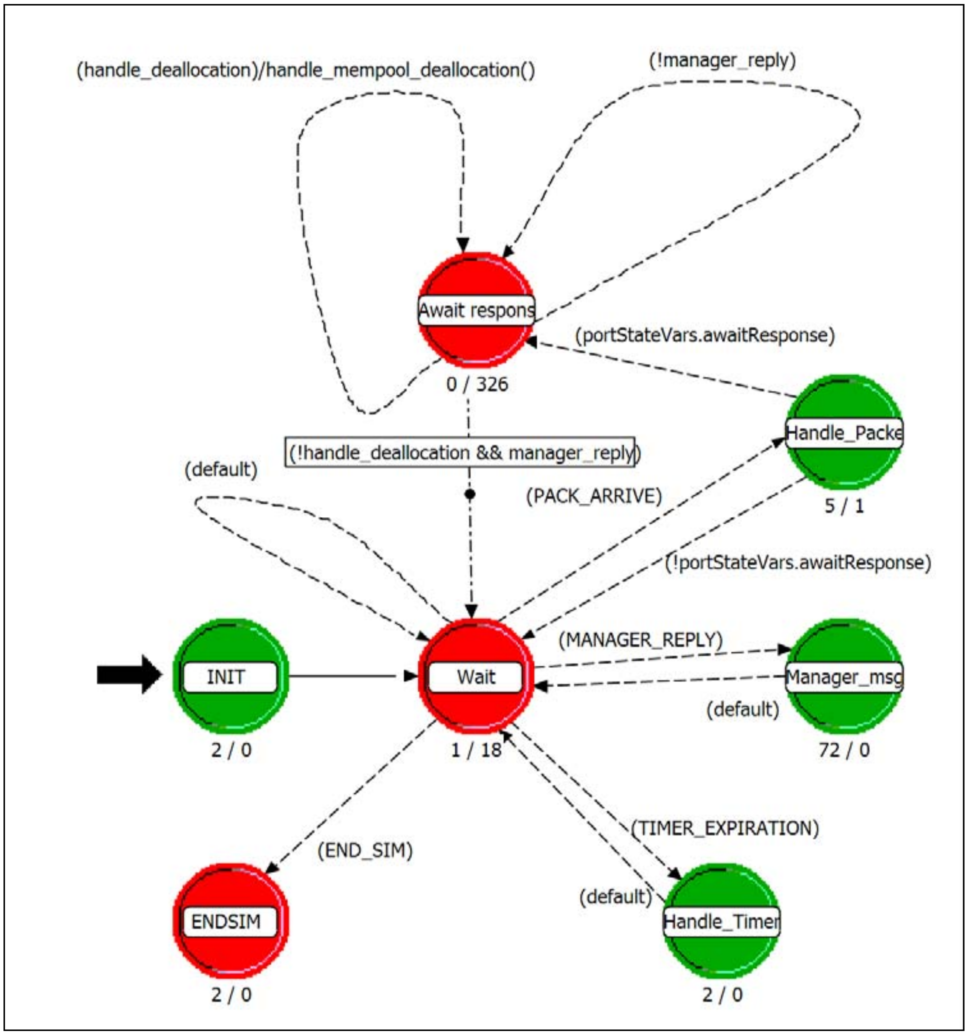

Figure 7. Proxy Process Model

In Figure 7, two new loop-backs are connected to the await response state. The loop-backs are manager deallocation and manager reply. The manager deallocation became necessary when the proxy request buffer space from the manager and the manager needs to reclaim memory from the same proxy before allocating. When the deallocation message arrived, at the proxy waiting for a manager response the proxy handled the deallocation, but returned to the wait state without storing the packet. This error only happens when the requesting proxy has to deallocate, before it is granted space. The second loopback keeps the proxy in the await response state when a packet arrives before the manager response is received. Since the manager can not send a forced response, other events can happen during the response time. It is a drawback of the model, as incoming packets can not be serviced while the proxy is waiting for the response. To determine the number of occurrences, a counter was added to monitor the number of unhandled packets. On average, 20-30 packets were forwarded without being handled during an entire run. These packets could be IACKs, ACKs, data packets, or virtually any IP traffic. If the proxy misses an ACK or IACK, the next IACK will clear the buffer of the missed packet. If the proxy misses a data packet, then it will be forwarded without buffering. The preceding proxy router will automatically resend the data packet if the packet is not IACKed. In either case the buffering protocol will eventually reconcile the missing packets. Therefore, 30 packets forwarded without buffering does not warrant a rewrite of the code.

Memory Manager

The router has a single entity responsible for buffer management called the Memory Manager. The design decision to have a single entity control the buffer space is discussed in [1]. Primarily, the manager needs to control the distribution of memory across all links. There are four proxy process models on links 8, 9, 10, and 11. Memory is distributed on a first come first served bases. Harmon’s [1] manager deallocates memory when updates are sent from the proxy to the manager. Changes were necessary to reclaim memory when higher priority flows arrive. Reclaiming memory from a link required the manager to send a packet with connections identifying information. If a proxy is in the await response state, then it handles the deallocation and returns to the await state. If the proxy is in the wait state, it will enter the manager msg state, deallocate the memory, and return to the wait state.

The proxy sends update packets to the manager immediately after the queue is emptied. The proxy update packet allows the memory to be redistributed to other flows, whether higher or lower priority. Letting each flow keep the allocated memory instead of returning it to the manager, actually performed better for each individual flow. The buffer space was available reducing the number of requests and contention. This process increased the efficiency of the manager. The drawback is that high priority flows retained unused memory, and the lower priority flows received no memory; thereby reducing the system to buffer space versus the number of flows. Decidedly, the unused memory should be returned to the free memory giving all priorities an equal chance until the memory is exhausted again.

Memory Scheme

Some of the characteristics of the memory manager required changes to incorporate the prioritization buffering scheme. The original manager handled requests by allocating 10 packets to each request until the free memory was exhausted. On a single outgoing link with two or three flows, every flow received at least 10 packets. The networks exercised in this thesis have more than one outgoing link buffering packets to create contention for the memory. They are also handling 88 flows, which would quickly deplete the free memory. Specific project models are discussed in Chapter 4, the following paragraphs will discuss the changes in the memory allocation.

There are 11 priority levels created for the experiments. The priorities are 1 being the highest priority and 10 being the lowest priority. Priority 0 is the control group for the experiment. The proxy never requests memory for priority 0 flows. Priority 0 will provide a comparison in the experiment to determine if priority buffering provides an advantage to no buffering. The other 10 levels of priorities provide a sanity check against pure chance.

Decisions on how to allocate memory to priorities is derived from a min-max algorithm. The manager searches for the minimum priority with at least one packet space to reclaim. When memory is depleted, the manager reviews the link records to find which flow has the lowest priority out of the 4 links. Additionally, the flow must have memory to de-allocate. Simply searching for the lowest priority without regard to memory allocated caused the manager to return with no memory to reclaim.

The notion of deleting a complete flow from a buffer took consideration. Since the flow is a lower priority, the higher priority flow should get the memory. It was debated on how to reclaim the memory. If a single packet is reclaimed from a lower priority should the first or last packet be removed? Either one could be vital, as neither have been IACKed. Assuming none of the packets reached the next proxy, there are two outcomes. If the first packet on the queue is removed, the oldest one, and it has not reached the next proxy, then the rest of the packets can not be delivered until the gap is filled. The proxy can not send the packets out-of-order or else they will cause a TDA.

Since the proxy IACKed this packet, the preceding proxy has most likely deleted the packet. The only way to fill the gap is to create an ACK for the previous packet, or wait for a TDA or RTO.

If selecting the last or newest packet on the queue, the older packets can be sent but the lost packet will be acknowledged three times at the sender before it can be resent again causing a TDA, or worst case an RTO. Both of these solutions cause the sender to reduce the congestion window, which in this scenario is an advantage to priority buffering. If the goal is to get the high priorities through a degraded medium, triggering a TDA or RTO is advantageous to the high priorities. Deleting a single packet from the queue requires extensive coding to update acknowledgements, sequence numbers and numerous other variables. With the intricacies of deleting a single packet, and knowing the flow will lose throughput anyway, it was more beneficial to delete all buffered packets from the flow and return all but two packet spaces to the free memory. The two spaces are granted to the requesting priority.

If a single flow is granted two packet spaces it will most likely request more space in the near future. Two packets are granted to the high priority flow and the rest are returned to the free memory. This procedure reduces the search time if the memory reclaimed is greater than two. The disadvantage is a lower priority requesting the space again causing the process to repeat. Since the free memory is only given out in quantities of 2, the lower priority will not grow as fast as the high priority when taking into account the TDA or RTO which will reduce the throughput.

When equal priorities contend for space it is granted a first come first serve basis. It would be illogical to force an equal priority to reduce its throughput and suffer a TDA. The manager can search across multiple links to find the lowest priority. Equal low priorities on two separate links are resolved by taking the flow with the most allocated memory. The reasoning is similar to reclaiming all the memory, if a flow is going to suffer a TDA, then the maximum benefit should be gained.

Creating Priorities

Considerations for prioritizing included adding bits to packets, setting flags in the TCP header, and adding information to the data of the TCP segment. All of these designs required changes in OPNET‘s TCP module. Changes to TCP were excluded, so the next option was to use existing information. The reasoning for not changing the TCP module is to limit the possibility of inducing artificial gains. The only information the buffing protocol is aware of is in the TCP header. Logically, the most efficient way to create a priority is using the information available such as port number.

Table 1. Priorities by Port Number

Priority	Ports	Priority	Ports
1	1000-1010	7	7000-7010
2	2000-2010	8	8000-8010
3	3000-3010	9	9000-9010
4	4000-4010	10	10000-10010
5	5000-5010	0	All Other Ports
6	6000-6010		

The port number was chosen to define a priority. There are 65,536 different port numbers available. The priorities are determined by taking the modulus of the port number subtracting it if falls between 0 and 10. The remainder is then divided by 1000 giving the priority. There are 11 available ports for each priority but this is easy to expand. The priorities are listed in Table 1. Choosing the port number as the priority indicator was a matter of simplistic design.

To generate the traffic in the Harmon model an application process model created a File Transfer Protocol (FTP) connection. A 20 MB file download is simulated. Configuring the FTP application to use a specific port is difficult and could change the TCP module to implement. Instead, for this research, a task configuration node is added to the project for manual task configuration. Each task has 11 phases, priorities 0-10 and all are configured to begin at the same time. A task differs from the FTP application in that the number of packets for the request and response are configurable. After a TCP connection is established, a single request packet spawns a configurable response of packets. The phase definition also allows the port number of the destination to be set for each priority. Disadvantages to this design happen when repeating the task. If the previous task does not close the TCP port, then the phase will fail stating the port is already in use.

Summary

This chapter described the model developed to simulate a realistic TCP aware link layer buffing device. This chapter reviewed the model implementation and the logic behind the model detail. Chapter 4 discusses the methodology of network design.

IV. Methodology

Chapter Overview

This chapter provides the details of the development of the network model and the experimental design. Following that is a discussion on the factors, parameters, and measurements used in the analysis of results in Chapter 5.

Simulation

As mentioned previously, the simulation tool used to implement the model is OPNET 14.0. The model is partly coded in OPNET and partly in external code, which allows for troubleshooting in Microsoft Visual Studio. In Visual Studio, breakpoints can be set and each method can be verified for proper operation during the running of the simulation in OPNET. Visual Studio provided direct access to the variables and structures which led to the changes discussed in Chapter 3.

Many small simulations were designed to test the model and to confirm proper operation. Although the networks presented in this chapter can not exercise every aspect, flavor of TCP, or attribute available in the simulation software, the experiments are designed to test the operation of the model and analyze the benefit of prioritizing buffered traffic. Time constraints limited the number of experiments; hence, many factors that are just as important in choosing to buffer network traffic were omitted. Presented in this chapter is a thorough review of a small scope of properties, with areas of future work presented in Chapter 5.

System Under Test

The simulation model constraints listed here focus the experiments to a specific set of characteristics. As stated in the constraints in Chapter 1, the model can not change the TCP software in the end point systems. OPNET offers a suite of configurable TCP versions. Therefore, one simulation challenge is determining what TCP settings should be used in the end points to reflect typical systems. All TCP versions contain the basic functionalities found in [6]. The differences represent optimizations and how they are affected by congestion mechanisms and throughput. The scope of this research was not to determine the specific TCP version that worked the best. However, the research can not ignore specific parameters which impede network performance when buffering is employed or the parameters which impede the buffering protocol.

The final project used the default TCP settings supplied by OPNET. The two changes used for all experiment scenarios are the maximum receive window size and the maximum acknowledgement response time. The default OPNET setting for the maximum receive window size is 8760 bytes, or 6 packets assuming 1460 bytes per packet. To exercise the revolving buffer, the default settings were changed to allow 44 packets in flight. The maximum time for a receiver to respond with an acknowledgement was reduced from 200 milliseconds to 1 millisecond. The buffering protocol uses the acknowledgement to clear the buffer and a local retransmission timeout (RTO) to resend buffered packets waiting for acknowledgements. The default setting of 200 milliseconds increased the number of resends from the proxy to the receiver. In an actual network the client may be far away, through many fixed networks. This would increase the RTO for the last proxy router, and therefore decrease the number of packet resends. In the simulations the client was directly connected to the proxy router.

Factors

The factors listed in Table 2 were specifically chosen after numerous designs were tested and implemented. Proxy buffering may present a benefit over a lossy network, but may impede TCP operation in a good network. To qualify an answer, many scenarios are necessary. Proxy buffering should provide a benefit not only to a small download size at low packet loss, but to a medium download size with high packet loss. Full factorial experiments were run with 30 different seeds, giving 540 experiments. Note that the priorities must be run at the same time to compete for the buffer space. Priority 0 is a control group which is simply forwarded without buffering. This provides a baseline in each experiment.

Table 2. Factors

<table>
<tr><th>Factors</th><th>Levels</th><th colspan="4">Description</th></tr>
<tr><td>Proxy Buffering</td><td>2</td><td colspan="2">Enabled</td><td colspan="2">Disabled</td></tr>
<tr><td>Packet Discarding</td><td>3</td><td>No Loss</td><td colspan="2">Low Loss 5%</td><td>High Loss 25%</td></tr>
<tr><td>Failure Intervals</td><td>1</td><td colspan="4">Exponential with 80 milliseconds mean</td></tr>
<tr><td>Priority Levels</td><td>11</td><td colspan="4">Priority 0 – 10</td></tr>
<tr><td>Message size</td><td>3</td><td>Small
(100 KB)</td><td colspan="2">Exponentially Varied
(mean of 512KB)</td><td>Medium
(1 MB)</td></tr>
<tr><td>Seed</td><td>30</td><td colspan="4">Multiple Random Variables</td></tr>
</table>

Message sizes are set by changing the number of response packets. (All packets are 1024 Bytes in data length.) The message size is chosen from an exponential distribution with a mean of 500 packets. Varying the download size is necessary to ensure that the packet sizes are not a factor in determining priorities. Larger downloads tend to gain higher throughput in low loss environments, where the smaller downloads complete quickly in the presence of high bandwidth.

When the simulation is exercised, all downloads begin at the same time to create competition for the buffer space and throughput. The tasks are set to repeat, with a constant delay between repetitions. The delay reduces the probability that a client will connect to a server on a port in use and cause a failure of two downloads. When an open command is received on a port already in use, a reset is sent to the download in progress destroying it. The client sending the open command does not receive a syn-ack and therefore after three attempts fails to complete.

Parameters & Measurements

The extensive abilities of simulation software can impede modeling. Missing parameters can affect the simulation and cause misleading results. Parameters for the OPNET TCP version are default parameters except for the two discussed above, unless otherwise noted. Building the simulated network required considerable tuning to provide enough competing traffic to demonstrate prioritized flows without exceeding the memory constraints of the hardware. These considerations guided the choices for link rate and buffer size. The link rate choice is a T-3 rate of 44.7 Mbps. This rate provided enough bandwidth to reach a steady state of 44 packets in flight. It permitted the download to complete in reasonable simulation and wall clock time.

The total buffer size per proxy router was set at 250 packets. The amount of used memory during the preliminary experiments rarely grew above 500 packets, so 250 packets (365 KB) was chosen to create a limited repository of available storage space.

Contention for memory space is required for a priority buffering scheme to operate. Table 3 summarizes the scenario parameters.

Table 3. Scenario Parameters

Parameter	Setting
Buffer Size (per Proxy Router)	250
Link Rate	T-3 (44.7 Mbps)
Number of Flows	> 90
Propagation Delay	Distance Base (<4 ms from end to end)

The statistics collected for determining the value of priority buffering is a combination of multiple results. The statistics listed in Table 4 give a description of the measurements of interest.

Table 4. Measurements

Measurement	Description	Units
Task Response Time	Time from connection establishment until the final packet is received at the client	Seconds
Throughput	Amount of traffic or packets traversing a particular link in a single direction	Bits per second or packets per second
Utilization	Proportion of link usage versus link capacity	Percentage 0-100%
Congestion Window Size	The boundary for the number of outstanding bytes [1]	Bytes
Number of Completed Downloads	The total times a single priority is downloaded per simulation period	Number

[1] Although the congestion window size is the upper bound of the number of outstanding bytes, the true number of outstanding bytes is the minimum of the congestion window and the maximum receive window.

The task response time is a measurement from the time the TCP connection is made and first request packet is sent until the last response packet is accepted at the destination. Task response time alone would not provide a good measurement of the prioritization since the last packet in a high priority download may be lost. When the last packet is lost, the high priority's task response time may be worse then a low priority based on probability. Averages of the task response time are reliant on the number of downloads and the actual time of completion. All downloads begin at the same time, increasing the delay in the system. The task response time increases proportional to the amount of traffic in the system. Therefore, additional downloads increase the task response time average.

Throughput is the measurement of bytes per second and in this research the measurement is taken as the traffic enters the client. Every client represents a single priority and all the TCP traffic is sent to one of the clients. The total throughput for the network includes other traffic such as routing information IACKS and ACKS. However that traffic is small in comparison to the actual file size traffic. The throughput prediction is that higher priorities should have the highest throughput, as they experience less loss. The number of high priority flows on a given link, the number of failures, and the number of hops all influence the throughput. These affect the RTT, and are inversely proportional to throughput. Lastly, the number of downloads can artificially increase the throughput for a client as there may be numerous concurrent downloads across multiple servers.

Network Modeling

In modeling the network it is essential to avoid misrepresenting results. Careful consideration to alleviate false results is presented here. Any network bottlenecks present preferential treatment to link or flows. Maximum utilization in the network may misrepresent the flow's throughput. Multiple independent runs confirmed having all the high priorities located together caused the reduction in a flow's throughput. Thus, the network was designed to distribute priorities. Ultimately, the priorities still must compete with each other for buffer space, so they still shared a common link in the central router shown in Figure 8.

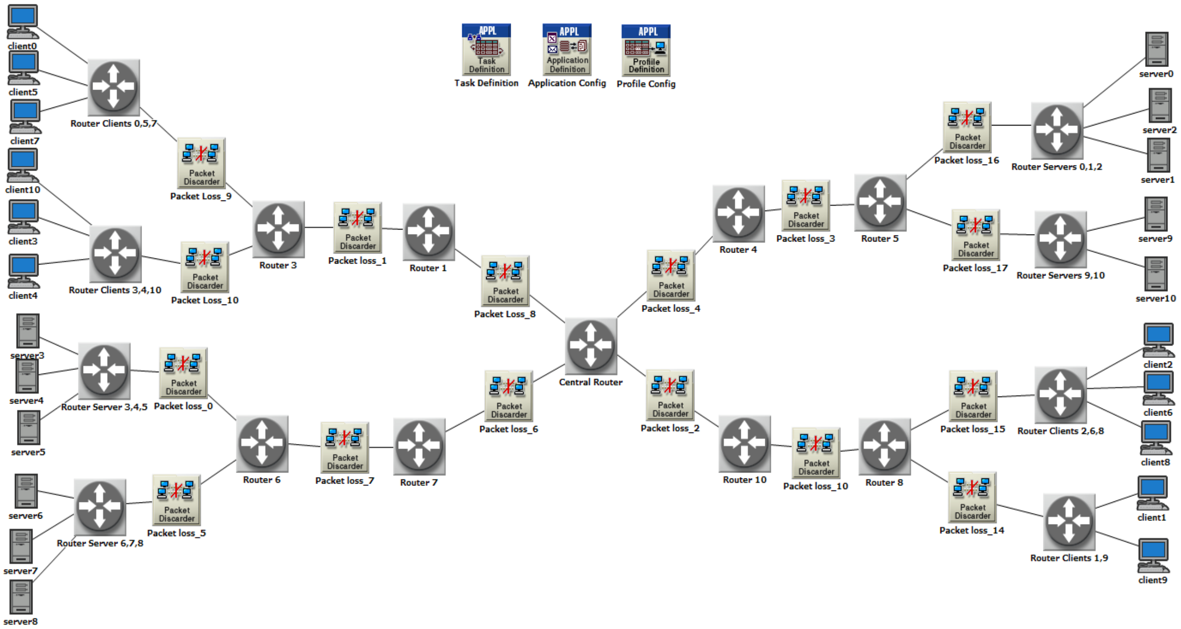

Figure 8. Final Scenario Design

Randomness in task execution order becomes vital. All the tasks begin at the same time, but they should be random. However, the simulation software must have some order in which to queue up the events, which may impose an undesired structure on the order. Design challenges limited the clients from each owning multiple priorities and instead each client is a priority. For randomness, the servers from which the clients download are chosen at random, and all servers can provide all priorities. Connection for a specific priority can be diverted to another server if a server is busy.

IV. Analysis and Results

Chapter Overview

This chapter reveals the outcomes of the simulations along with the knowledge gained from the analysis of the results. In this chapter, the investigative questions are answered, and the data gathered during the simulation is presented for the reader's assessment.

Experimental Questions

The goal of the research presented here is to find the answers to the experimental questions. The experiments test the additional features implemented in the model and their affect on the network. The changes should not induce a negative effect on the network. The effects of prioritizing network traffic are explored, including potential benefits, and provide answers to the following experimental questions.

Experimental questions:

1. What are the overall effects of prioritized buffering on the network?
2. Does limiting the number of buffered packets to 44 reduce or limit the throughput of the link?
3. Do the algorithms for determining a priority's buffer space favor higher priorities?
4. How does link layer buffering react to routing changes?
5. Does an exponential packet discarding interval distribution provide a realistic degraded environment?

6. If buffering conceals the actual congestion, what is the overall effect on the fairness in the absence of loss?

Results

Analyzing the results in Figure 9 indicate there is an observable throughput benefit to larger files. Large files have the ability to gain momentum, or the ability to gain higher throughput over time. This effect is accomplished in three ways: decreasing the RTT, increasing the congestion window size, and increasing the number of incoming acknowledgements. The RTT in this scenario has a fixed minimum, but may increase as a result of packet loss and subsequent recovery. Since the packets are buffered, if they are delivered in such a manner to avoid an RTO, then the RTT may slightly increase. The congestion control window is an indicator of momentum, but the measurement is throughput, and after the maximum receive window is reached flow control limits the packets in flight. Each incoming acknowledgement received without an RTO or triple duplicate acknowledgement increments the sliding window. The rate of incoming ACKs determines the rate at which the sliding window moves. This rate is the momentum of the throughput. When an RTO event occurs, the window is stopped, and the process begins again in a slow start state which takes time to reach the high momentum.

Smaller files lack the ability to gain momentum because they are completed before they reach the congestion avoidance phase. The smaller files used in the experiment are 100 KB. Analysis indicates they easily complete before reaching the 44 packet window. The medium files are 1 MB in size or 1000 packets, and have the ability to reach the theoretical maximum throughput before completion. The varied file size is an exponentially distributed file size with a mean of 512 KB. The results in Figure 9 provide the throughputs for large, varied and small files.

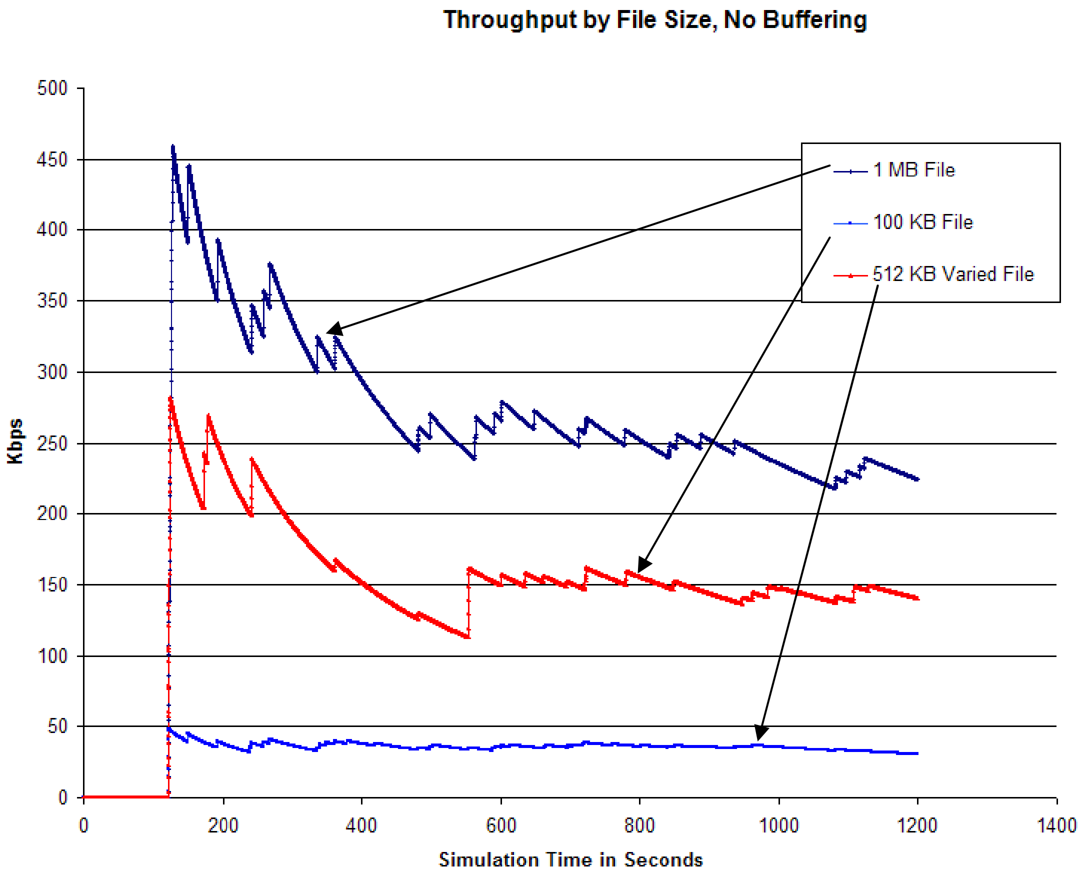

Figure 9. File Size Average Throughput Comparison with No Loss, Buffering Off

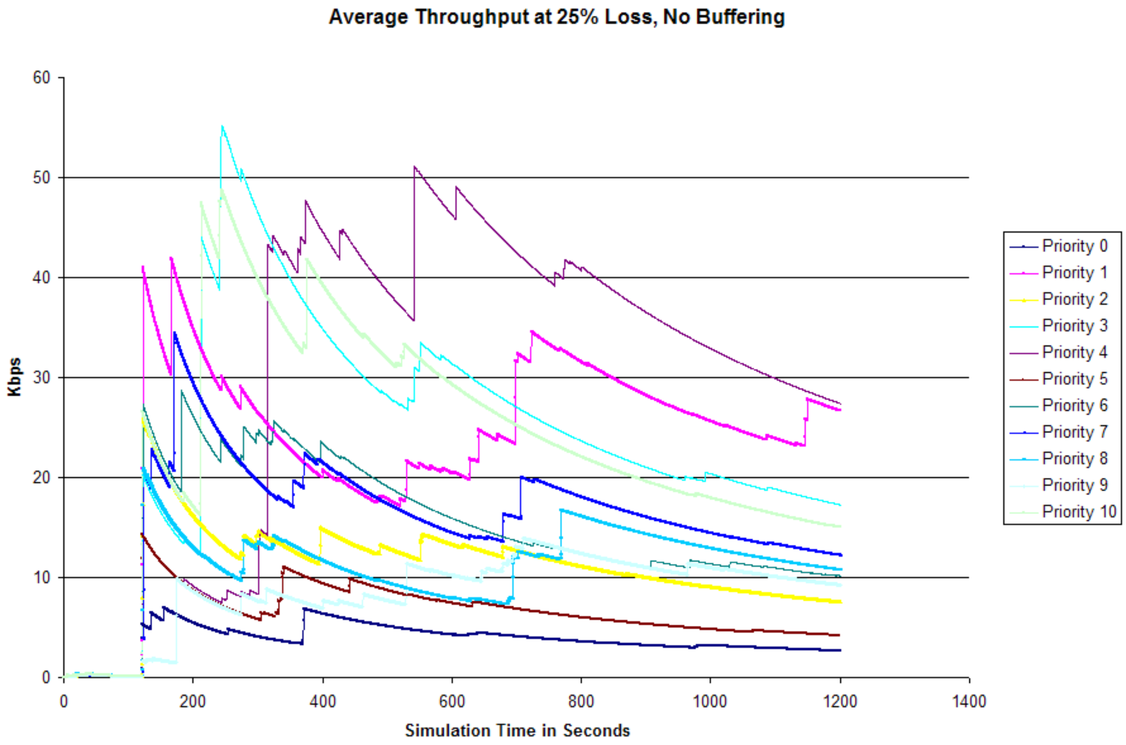

Figure 10. 25 Percent Loss, Priority Buffering Off, Large File size

Figure 10 shows the average throughput for all 11 clients, in a network with 25% loss per link, 1 MB file size and buffering off. The average throughput peaks at 55 Kbps, and the stead state mean is 13.5 Kbps. In comparison, Figure 11 illustrates the same parameters with buffering on. The average throughput peak of the traffic is 500 Kbps, and the steady state mean is 80 Kbps. It is noted that priority zero is not buffered therefore it is excluded from the steady state mean. The average steady state throughput is 6 times greater when buffering is applied.

What is significant to priority buffering is that priority 7 has claimed the momentum and is above the throughput for the other clients in steady state. In Figure 11, the final average throughputs of priorities, in descending order, are: 7, 6, 3, 9, 1, 8, 5, 10, 2, 4, 0. If priority 1 is the most important why doesn't it have the greatest average throughput? The problem is priority 1 is not utilizing the buffer space. Even though it has space available to it but because it suffered packet loss, or real congestion at the first router, it is not building enough momentum use the buffer space which could be allocated to it.

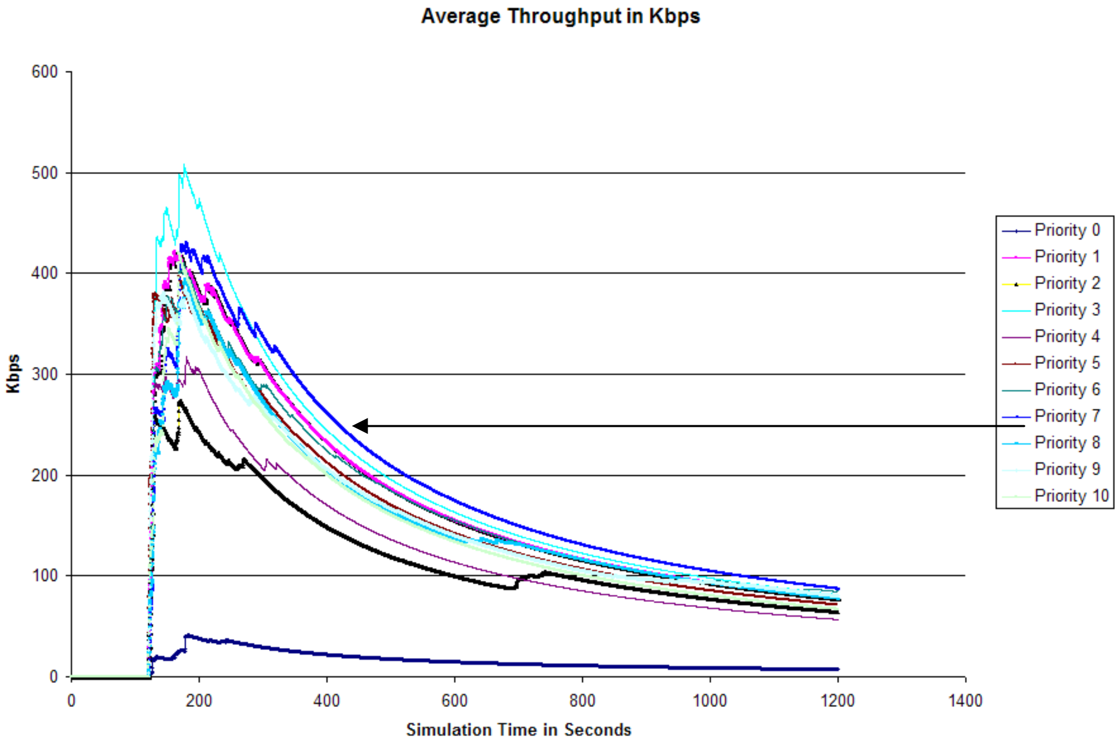

Figure 11. 11 Priorities at 25 Percent Loss, Priority Buffering On, Large File Size

When priority buffering is applied to a low-loss situation, there exist the potential for one TCP connection to gain momentum and saturate the link. Figure 12 illustrates a case with buffering on, using the congestion window as an indicator. This chart was developed by taking the top 10 maximum congestion window sizes for a single router. In Figure 12, the connection from Server 3 to Client 5 has a maximum congestion window 100 times larger then the next closest maximum, noting that the scale is logarithmic. The average congestion window size in Server 4 to Client 0 is sufficiently lower than the rest of the averages suggesting that the incoming link to the "Router-Clients 0, 5, 7" saturated by the client 5 connection. When this occurs, real congestion losses cause Client 0 to lose throughput. As indicated by the average over the total download, the Server 4 - Client 0 connection had one packet in flight on average.

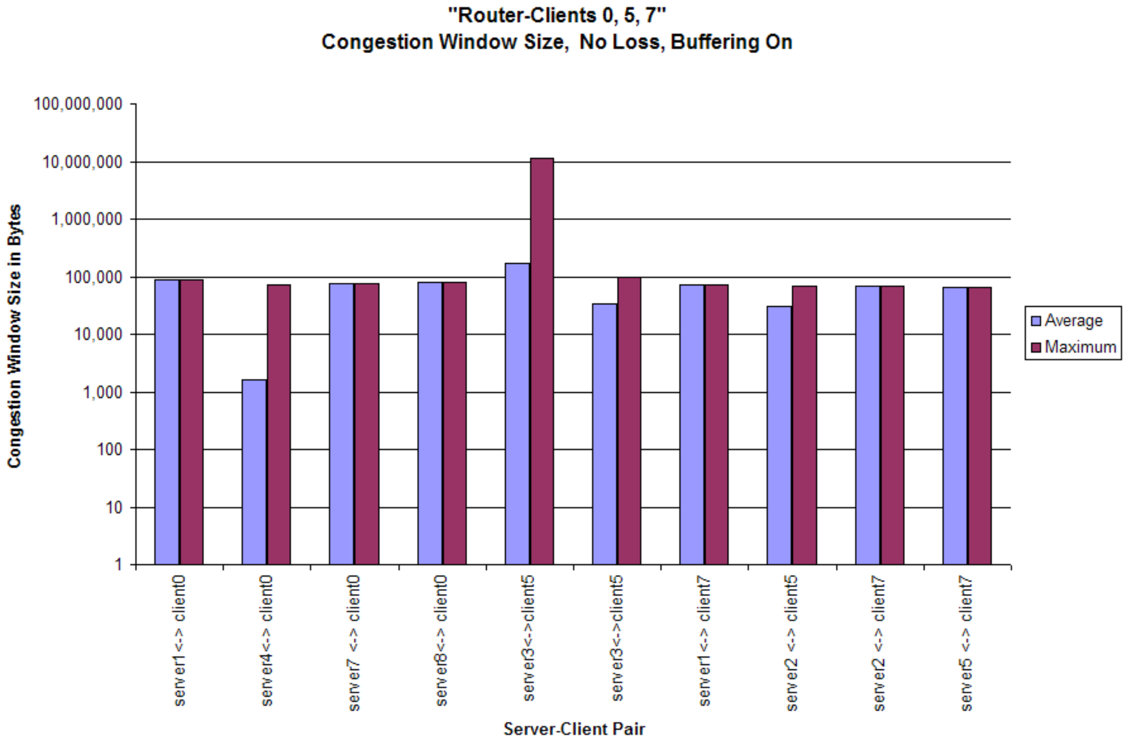

Figure 12. No Loss, a Large Files size, and Priority Buffering On, Logarithmic scale

Figure 13 shows the same experiment with buffering off; noting it is not a logarithmic scale. The congestion windows all share equitable maximum and average congestion window sizes. As shown in Figure 14, the throughput for both of the experiments indicates buffering provides unfairness. Client 5 has a higher average throughput in both experiments. The average throughput for Client 0 and Client 7 is significantly lower when buffering is applied. The priority buffering allowed Client 5 to gain momentum and flood the link, leaving less throughput for Client 0 and Client 7.

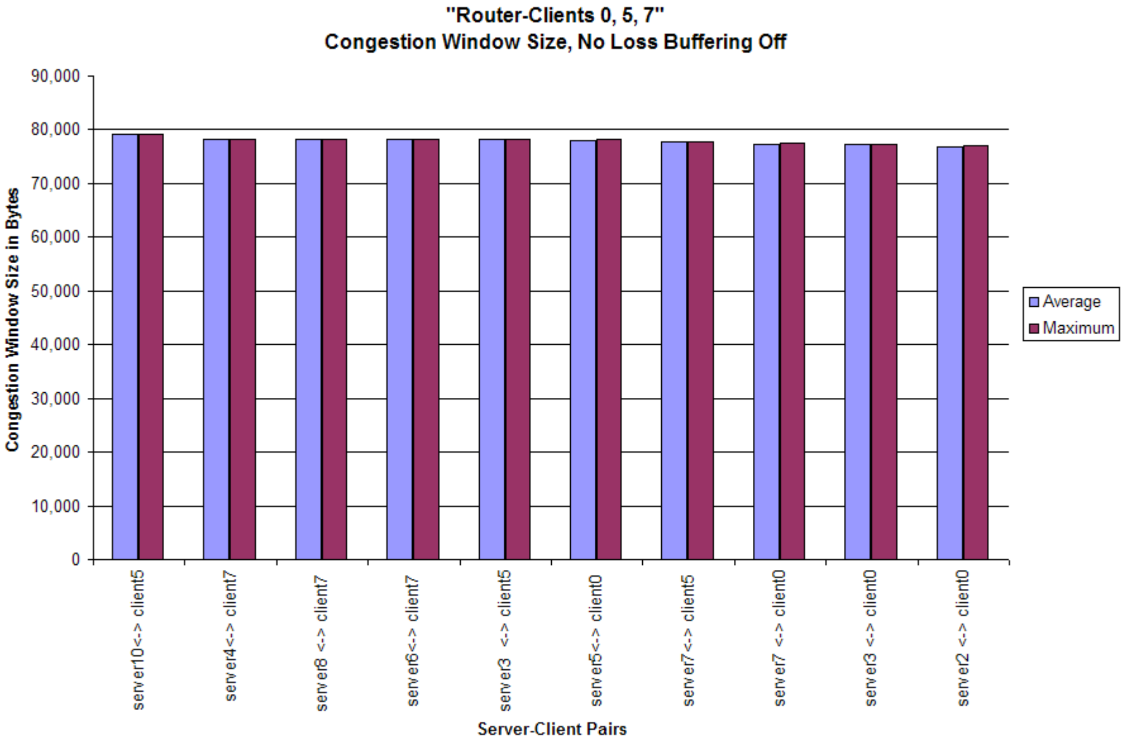

Figure 13. No Loss, a Large Files size, and Priority Buffering Off

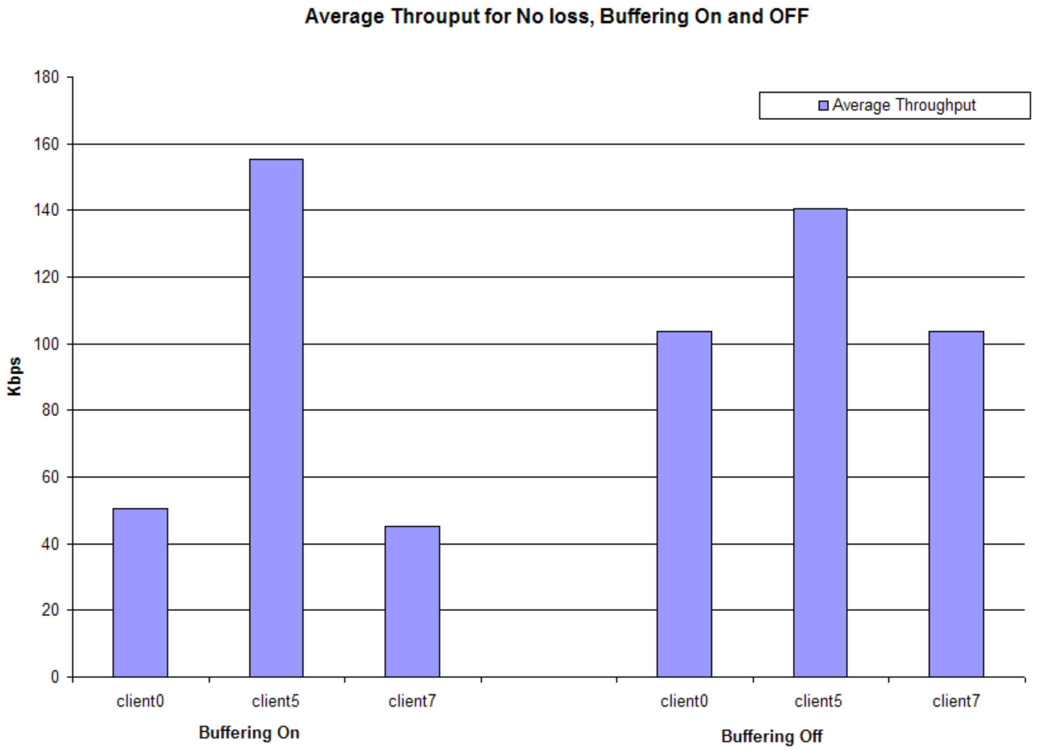

Figure 14. Average Throughput Buffering On an Off in No-loss Environment

Investigative Questions Answered

Question 1. What are the overall effects of prioritized buffering on the network? Priority buffering provides a significant factor of six increase in average throughput. What it's missing is the guarantee to bolster the high priorities over the low ones. The system can not predict the packet recovery time so it still suffers RTOs. Buffing provides a reliable delivery of packets in the degraded network. In a 25% loss environment buffering provides four times the throughput versus not buffering. While prioritizing flows can not increase the throughput of a higher priority it does provide an increase in throughput when the priority gains the momentum necessary to use the buffer space provided to it. Control of the intricate timing necessary to implement a fully operational prioritized buffer provides the potential for additional research.

Question 2. Does limiting the number of buffered packets to 44 reduce or limit the throughput of the link? The amount of buffer space needed for the experiments run was always less than 500 packets per router. The maximum amount of outstanding packets for any one flow is 44. The average number of packets need per flow depends on the momentum.

The experiments reached a maximum of 99% utility of the links provided, and the buffer allocated per flow never filled the 50 packet capacity, which it shouldn't since the maximum receive window is set to 44 packets. The RTT for the IACKs is much smaller then the actual data packets so the IACKs from proxy to proxy immediately clear the buffer before they have time to grow.

Question 3. Do the contention algorithms for determining a priority's buffer space favor higher priorities? Priority 1 flows are always granted space in the buffer. Priority 10 flows are always the first to be removed from the buffer. This effect creates a hidden benefit to the priority 1 flow by forcing a congestion event on the priority 10 reducing the throughput of 10. Priorities such as 4, 5, and 6 receive buffer space and are not affected by the prioritization unless a very small limit, around 150KB, is placed on the overall buffer size.

There may be additional benefits of reordering the outgoing packets to send the high priorities first. This modification would be an intricate design where packets enter on two different links are handled by precedence and not on a first come first serve basis. Additional design changes may extract the high priorities from the incoming queue first but this presents unique and undiscovered problems.

Question 4. How does link layer buffering react to routing changes? To investigate this question, a scenario was created to change the route from client to server during a FTP download as discussed in Chapter 3, shown in Figure 5. The addition of the revolving buffer fixed the gap problem and the FTP download was completed. The routing required the OSPF 'dead-link' to be reduced an unrealistic value. The buffers that are disconnected from the flow are able to clear the buffers after a predetermined time.

Question 5. Do changes the packet discarding interval to an exponential distribution provide a realistic degraded environment? The packet discarder or link winker created for previous research worked on a fixed interval size. Downtime will always be a multiple of the fixed interval. For this research, the interval is varied to mimic a real network experiencing scintillation effects, or interference. The interval of 80 milliseconds was chosen to be magnitudes greater then the RTT time between routers and greater then the end-to-end delay. The discarding times are exponentially distributed, providing many small intervals and a few large intervals. The original packet discarder may have provided a predictable throughput, which TCP adapted to compensate for the loss in the RTO smoothing algorithm.

Question 6. If buffering conceals the actual congestion, what is the overall effect on the fairness in the absence of loss? Buffering minimizes packet loss in a degraded environment but it also hides packet loss in a low-loss environment. This becomes apparent when looking at the congestion window size. The amount of outstanding packets, or packets in flight, is the minimum of the congestion window or maximum receive window. The congestion window growth is a good indicator of fairness when multiple flows are sharing the same link. If all downloads begin at the same time, have the same delays, and experience no loss then they should all have an equal share of the link. If all windows grow at the same rate then they are each receiving acknowledgements at the same rate. As indicated in Figure 12, the congestion window of the Server 3-Client 5 connection is well above the rest. Turning the buffers off and running the same experiment showed the equality of the congestion windows average and maximum. Congestion window growth without bound is indicative of zero loss, which is the goal of buffering the packets. The number of connections in the experiment remains the same, but there are more connections with low throughput while a few connections are clearly gaining an unfair share of throughput.

Summary

This chapter reveals facts about priority based buffering using a link layer TCP aware protocol. The analysis of the results shows a benefit to the higher priority flows, if the flows are not affected by a retransmission timeout (RTO). Once the throughput is reduced, the benefit of buffering diminishes. Priority based buffering provides a clear answer to link layer buffering in large capacity networks where the overhead of buffering every TCP packet would not be feasible. In this case, a small percentage of traffic with a priority could be buffered while all other TCP traffic is transmitted without buffering. Chapter 6 provides a look at future work that may benefit from this protocol.

V. Conclusions and Recommendations

Chapter Overview

Chapter 5 presents the conclusions drawn from analysis of the Chapter 4 results. The research presented here revealed interesting challenges that lie ahead in the field of link layer buffering. There are still hurdles to overcome, however, priority buffering is feasible and provides an intermediate fix to TCP in a wireless domain. Last, recommendations are presented for future work in this area.

Conclusions of Research

There are differing opinions on the worth of network buffering [7, 11]. Opposing views to link-layer buffering discuss the overhead required for processing, storage, and memory access outweighing the benefits of buffering. These views place emphasis on changing the TCP end-points to control the flow, so the user has the control authority of the TCP. Further research describes TCP as misbehaving in a wireless environment. This work concludes that TCP is behaving appropriately, and it is the challenged network which requires improvements to increase the performance. It is the authors' perspective that the challenged network is causing the degradation, and requires adaptable controls to reduce loss and present a transparent medium to the end-points.

This research provides a clear understanding of the benefits and drawbacks of a priority based buffering system. Under some conditions, buffering without prioritization provides a four fold increase in network throughput in a degraded network. Prioritization of the traffic is still missing the key control element that is necessary to maintain a steady state condition. That key element is the ability to predict and control flow rates. This element must be able to adapt to network outages and predict an amount of delay to induce to smooth the retransmission time out algorithm. As long as a packet loss recovery time is greater than the retransmission timer, the throughput will be reduced multiplicatively.

Military networks are designed to be robust and redundant, and in the presence of adversity, buffering may provide increases in throughput, utilization, and connection ability. The challenge is to condition the buffer so it predicts the nature of the packet loss and provides a steady state RTT for the end-point server.

Prioritization of packet flows provides a boost to the higher priorities. As seen in Chapter 4, priority based buffering relies on factors such as the connection establishment, throughput, packet loss recovery, and the specific packets lost. TCP does not provide a steady flow of packets when retransmission timeouts occur. When a single flow’s traffic is reduced, it no longer has a buffering advantage. The prioritized buffering protocol favors the flow with the most throughputs, unless the buffer space is limited. If the flow does not ask for buffer space it will be given to the flows with a need. This effect limits the ability prioritization, and therefore is an area of future research.

Limiting the buffer space per flow showed that 44 packets are sufficient when windows scaling is not employed. Windows scaling can have up to 1 GB of data in flight which would need at a maximum 735K packet spaces per flow. Determining the buffer space may prove to be a tough challenge, as generating enough traffic in a simulation to fill 1GB of data per flow may not be feasible.

Significance of Research

This research provides a means to improve the performance of challenged networks. As military systems incorporate more technology, the ability to create a system of systems relies heavily on the line of communication. As more complexity, or more entities are added to the system the more throughput and reliability is required. The research presented here is significant in that it explores an intermediate fix to a protocol that needs redesigned to encompass a network loss. Since employing a new wireless TCP design is not feasible at this time, buffering provides a workaround until that day. Prioritization is an addition to buffering that limits the amount of memory and processing delays required to buffer every packet in a degraded network.

The research explored the ability of a TCP-aware priority based buffering protocol, and revealed the benefits and implications of employing such a network. There are numerous factors to consider when designing such a system and this document provides a roadmap to fulfilling that goal. The findings in this document provide a stepping stone to the next iteration of link layer buffering. The ideas discussed in the future work section will enlighten one to the possibilities of increasing the capabilities of a degraded network without disturbing the TCP structures in use.

Recommendations for Future Research

TCP provides a difficult task of timing the link’s downtime. More work in TCP congestion control timing is necessary to compliment this research. The link’s downtime provides a crucial role in throughput of the high priority traffic. As shown in the results of Chapter 4, when the packet loss and recovery take longer then RTO the benefit of buffering is lost. The next iteration of this research should build upon the ideas presented and incorporate some type of link downtime prediction software. If the proxy router can predict or calculate the link's downtime then packets departing on this link can be delayed. This delay will produce a larger RTT which is used to calculate the RTO.

A controller of the packet flow through degraded network is necessary to decrease the packet loss and increase the overall throughput. The controller needs the ability to measure network outages and predict future delays.

Since buffering is providing the reliability in the system, the second path of this work would incorporate the User Datagram Protocol (UDP). UDP has no congestion mechanism to overcome, and reliability is built into the priority buffering protocol. TCP limits the traffic when affected by adverse conditions, where UDP will continuously provide the throughput under any conditions. Two challenges with UDP will be overcoming the throughput of UDP and requesting packets lost outside the system. UDP is limited only in the processor's ability to generate the traffic and the bandwidth imposed by the link.   If the UDP end-points are outside the system then providing a way to resend missing packets is a difficult task which will take some consideration. There are applications that provide piecemeal tracking, such as bit torrent, which would allow the end-point to determine the missing data and resend just that particular segment.

Summary

The research presented here provides a preliminary look at priority buffering of traffic in a challenged network.   The analysis shows that given a network with high packet loss, prioritized traffic can be favored over low precedence as long as the throughput of the flow is not disturbed.   For buffer prioritization protocol to work the traffic generator can not be impeded by congestion control mechanism. Congestion control in a wireless network provides still an interesting problem; however, with the knowledge gained in this thesis there may be an answer to increasing throughput in a challenged network.

Bibliography

1. Harmon, D.F., *Overcoming TCP Degradation in the Presence of Multiple Intermittent Link Failures Utilizing Intermediate Buffering*, in Department of Electrical and Computer Engineering. 2007, Air Force Institute of Technology: Wright-Patterson AFB.

2. Stevens, W.R., *TCP/IP Illustrated Volume 1*, August 2004 - 25th Printing: Addison Wesley, Reading MA.

3. Reynolds, M.B., *Mitigating TCP Degradation Over Intermittent Link Failures Using Intermediate Buffers*, in *Department of Electrical and Computer Engineering*. 2006, Air Force Institute of Technology: Wright-Patterson AFB.

4. Balakrishnan, H., et al., *Improving TCI/IP Performance over Wireless Networks*, in *Proceedings of the 1st annual international conference on Mobile computing and networking*. 1995, ACM Press: Berkeley, California, United States.

5. Mathis, Matthew, et al., *The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm,* Computer Communication Review, volume 27, number 3 July 1997

6. Postel, J., *IETF RFC 793: Transmission Control Protocol*. September 1981.

7. Fall, K. *A Delay-Tolerant Network Architecture for Challenged Internets*, SIGCOMM'03, Aug 25-29, 2003, Karlsruhe, Germany

8. Fieger, A., Zitterbart, M., *Transport Protocols over Wireless Links*, ISCC '97: Proceedings of the 2nd IEEE Symposium on Computers and Communications

9. Hacker, T., Nobel, B., Athey, B., *Improving Throughput and Maintaining Fairness using Parallel TCP*,INFOCOM 2004, 23rd Annual Joint Conference of the IEEE Computer and Communications Societies

10. Buchholez, G., Ziegler, T., Do T., *TCP-ELN: On the Protocol Aspects and Performance of Explicit Loss Notification for TCP over Wireless Networks,* 1st International conference on Wireless Internet, WICON 2005

11. Balakrishnan, H., et al., *A Comparison of Mechanisms for Improving TCP performance over Wireless Links*, ACM SIGCOMM '96, Stanford, CA, August 1996

12. Scott, J., Mapp, G., *Link Layer-Based TCP Optimisation for Disconnecting Networks,* ACM SIGCOMM Computer Communication Review, Vol 33, Num 5, October 2003

13. Habib, A., Bhargava, B., Fahmy, S., *A Round Trip Time and Time-out Aware Traffic Conditioner for Differentiated Services Networks*, Proc. IEEE International Conference on Communication (ICC), New York, April 2002

14. Chiueh, T., Prashant P., *Cache Memory Design for Network Processors*, HPCS'00 Conference Proceedings, 2000

15. Miyake, Y. et al., *Acceleration of TCP Throughput over Satellite-base Internet Access using TCP Gateway*, Computers and communications, 2000, Proceedings ISCC 2000, 5th Symposium

16. Khan, F., et al., *Link Layer Buffer Size Distributions for HTTP and FTP Applications I an IS-2000 System*, Vehicular Technology Conference, 2000 IEEE VTS-Fall VTC 2000, 52nd Vol 2.

17. Bakre, A., Badrinath, B., *I-TCP: Indirect TCP for Mobile Host*, In Proc. 15th International Conf. on Distributed Computing Systems (ICDCS), May 1995

18. Batra, S., Chu, Y., Bai, Y., *Packet Buffer Management for a High-Speed Network Interface Card*, Computer Communications and Networks, 2007, ICCCN 2007, Proceedings of 16th International Conference

19. Fall, K., Floyd, S., *Simulation-based Comparisons of Tahoe, Reno, and SACK TCP*, Computer Communication Review, vol. 26, pp. 5--21, July 1996

20. Kopparty, S., et al, Split TCP for Mobile Ad Hoc Network, Proceedings of IEEE GLOBECOM, Taipei 2002

21. Robertazzi, T., *Computer Networks and Systems Queuing Theory and Performance Evaluation*, 3rd Edition, Springer, New York, 2000

22. Kurose, J., Ross, K., Computer Networking, A Top Down Approach Featuring the Internet 3rd Edition , Pearson Education, Inc., 2005

23. Weeks, M., Student Intern, Southwestern Ohio Council for Higher Education, Dayton OH. Personal Interviews. Sept 2007 – Feb 2008

24. Border, J. et al., *IETF RFC 3135: Performance Enhancing Proxies Intended to Mitigate Link-Related Degradations*, 2001

Vita

Kevin J. Savidge was born and raised in Bellmawr, New Jersey and entered the military in 1994. After basic he attended technical training in electronic and switching systems in Keesler AFB Mississippi. After which, he was stationed at Offutt AFB Nebraska, and worked on secure networks. He attended classes at University of Nebraska at Omaha, and earned an Associates Degree through the Community College of the Air Force. He was selected for Airman Education and Commissioning Program and attained a Bachelor of Science in Electrical Engineer and Computer Engineering from North Carolina State University specializing in embedded systems and networking. After commissioning as a Second Lieutenant, he was stationed at Tinker AFB, Oklahoma in the B-1 avionics engineering department. After being promoted to First Lieutenant, he was selected to attend the Air Force Institute of Technology where he was promoted to Captain and pursed his Masters of Science in Computer Engineering with Networking specialization. Upon graduation he will be assigned to AFOTEC at Nellis AFB, Nevada.

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 074-0188*

1. REPORT DATE *(DD-MM-YYYY)* 27-03-2008	**2. REPORT TYPE** Master's Thesis	**3. DATES COVERED** *(From – To)* Aug 2006 – March 2008

4. TITLE AND SUBTITLE Priority Based Buffering over Multiple Lossy Links Using TCP Aware Link Layer Buffering	**5a. CONTRACT NUMBER**
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Savidge, Kevin J., Captain, USAF	**5d. PROJECT NUMBER** ENG 08-175
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S) Air Force Institute of Technology Graduate School of Engineering and Management (AFIT/EN) 2950 Hobson Way, Building 640 WPAFB OH 45433-8865	**8. PERFORMING ORGANIZATION REPORT NUMBER** AFIT/GCE/ENG/08-10
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Office of Scientific Research Bonneau, Robert J AFRL/AFOSR 875 N Randolph Street Ste 325 Rm 3112 Arlington, VA 22203-1768 (703) 696-6565 (DSN : 426-6207), e-mail : robert.bonneau@afosr.af.mil	**10. SPONSOR/MONITOR'S ACRONYM(S)**
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT Wireless military information systems require high reliability, which is difficult to achieve in adverse conditions. To provide high reliability one must overcome packet loss across multiple wireless hops. Buffering packets in a lossy environment is well explored; however, the ability to selectively buffer TCP traffic across multiple lossy links is a new area of research. This document seeks to explore the delivery of high priority traffic in a lossy environment and conclude that prioritized buffing can increase the probability that a high priority download will finish, where others will fail.

It is shown that buffering provides six times the throughput in a network with each link experiencing 25% loss. Prioritizing TCP packet flows provides a varied outcome, as it can not overcome the TCP mechanisms, when the packet loss recovery time is greater than the retransmission timeout event. However, the future work in chapter 6 may provide roadmap to gaining control authority of the challenged network.

15. SUBJECT TERMS

TCP, Priority Buffering, Mobile Networks, Link Layer Buffering

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** Scott R. Graham, Maj, USAF (ENG)
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			**19b. TELEPHONE NUMBER** *(Include area code)* (937) 255-6565, ext 4918 (sgraham@afit.edu)
U	U	U	UU	82	