Wireless Network Enhancements Using Congestion Coherence, Faster Congestion Feedback, Media Access Control and AAL2 Voice Trunking

DISSERTATION

Presented in Partial Fulfillment of the Requirements for

the Degree Doctor of Philosophy in the

Graduate School of The Ohio State University

By

Chunlei Liu, B.Sc., M.S.

* * * * *

The Ohio State University

2001

Dissertation Committee:

Professor Raj Jain, Adviser

Professor Ten-Hwang Lai

Dr Arjan Durresi

Approved by

Adviser

Department of Computer and Information Science

Raj Jain is now at
Washington University in Saint Louis
Jain@cse.wustl.edu
http://www.cse.wustl.edu/~jain/

ABSTRACT

Rapid development of wireless communications in recent years has imposed great challenges for network support. As most current networking protocols were designed mainly for wired networks, many assumptions that make these protocols efficient in wired networks are no longer true and cause severe performance degradation in wireless environment. An urgent task for today's networking research is to identify such deficiencies and to enhance these protocols.

This dissertation contains four major enhancement results. The first result is a wireless TCP enhancement scheme called *Congestion Coherence*. Through a statistical analysis, we find the leading contributor of wireless TCP performance degradation is the timeout resulting from frequent packet losses. This scheme uses ECN to reduce number of timeouts, and based on the observation that congestion neither happens nor disappears suddenly, uses the sequential coherence of packet marking to determine cause of packet losses. Simulations show this scheme works better than existing enhancements and produces good performance.

The second result is a *Mark-Front Strategy* to deliver faster congestion feedback in networks that use the newly standardized ECN protocol. By choosing to mark the packet in the front of the queue, Mark-Front Strategy delivers faster feedback than traditional congestion feedback mechanisms. Our analysis show Mark-Front Strategy requires smaller buffers, and yields higher throughput and better fairness.

A MAC protocol to supports real-time applications in an asymmetric and half-duplex environment is our third result. Several innovative features are used to ensure optimal utilization of the asymmetric bandwidth.

In the last result, we use a Markovian chain to model the AAL2 voice trunking process and analyze the tradeoff between delay and bandwidth efficiency. This analysis gives a guideline for setting the packing timer. The result is adopted by a 3GPP industrial standard.

This study provides insight into congestion control and quality of service in wireless networks. The proposed enhancements will help to bring high-performance and high-quality services to mobile subscribers.

ACKNOWLEDGMENTS

Near the end of my student journey, I realize the years at Ohio State University have been a special and precious time in my life. Things I have experienced and learned here will have a profound impact in my future life and career. The completion of my study has been made possible through many people's support. I want to take this opportunity to express my sincere appreciation to them.

My adviser, Professor Raj Jain, has been a source of inspiration throughout the past four years. His broad knowledge and excellent teaching were the first motivation of my PhD study. His keen insight, amazing diligence and thoroughness in work have been and will continue to be my exemplar. I want to thank him for bringing me into the fascinating field of computer networks and providing generous financial support during my study.

I am very grateful to my dissertation committee members, Professor Steve Lai and Dr Arjan Durresi, and my candidacy exam committee members, Professor Wu-chi Feng and Professor Jennifer Hou. They have carefully examined my dissertation proposal and provided valuable feedback. Professors Roger Crawfis, Michael Fitz and Stuart Zweben have also been very helpful throughout my study. I would also like to thank staff members in CIS department, especially Elley Quinlan, Tom Fletcher, Elizabeth O'Neill, Marty Marlatt and Sandy Hill.

My parents have been extremely supportive of my academic pursuit. They have always stood behind me in difficult situations and given me encouragement. Without their education and support, I could not have come to where I am.

My colleagues at the Networking and Telecommunications Laboratory and good friends in CIS department, Sohail Munir, Mukul Goyal, Wei Sun, Sonia Fahmy, Bobby Vandalore, Rohit Goyal, Ming Liu, Yingjie Li, Ming-Te Sun, Mukundan Sridharan have given me a lot of help throughout my research. Their help in discussing ideas, solving computer problems and proofreading papers has been invaluable.

The completion of my dissertation was also encouraged by my colleagues at Lucent Technologies. I appreciate the encouragement and support from my managers David Rouse and Doug Moreland, and my office mate Lung Liang.

Rich Mendola, Anthony and Kimberly Padgett, Tom Thomin, Mabel Bahler, Virginia Reynolds and many other friends in Columbus have been part of my life. Their abundant help and encouragement can never be over estimated. Anthony Padgett, Frank Scott and Laura Weber proof read the draft of this dissertation and corrected many errors.

My thanks also go to my sister-in-law Kuei-Mei, who came to our home since June and has helped in many ways.

Finally I want to give special thanks to my wife Lih-Mei. Her love, patience and consideration have been the lasting support of writing this dissertation. I feel deeply indebted to her for the work she did for our family and for our newborn daughter Joyce. Without all these, this dissertation would have been impossible.

VITA

1968	Born, Puding, Guizhou, China
1988	B.S. Computational Mathematics, Wuhan University, Wuhan, China
1991	M.S. Computational Mathematics, Wuhan University, Wuhan, China
1991-1995	Lecturer, Wuhan University, Wuhan, China
1997	M.S. Applied Mathematics, The Ohio State University
1997	M.S. Computer and Information Science, The Ohio State University
1998 – 2000	Graduate Teaching and Research Associate, The Ohio State University
2000 – present	System Architect, Lucent Technologies

PUBLICATIONS

Research Publications

Chunlei Liu and Raj Jain. Improving Explicit Congestion Notification with the Mark-Front Strategy, *Computer Networks*, Vol 35, no 2-3, pp 185-201, February 2001.

Chunlei Liu, Multimedia Over IP: RSVP, RTP, RTCP, RTSP, book chapter in: *Handbook of Communication Technologies: The Next Decade* (editor Rafael Osso), CRC Press, Boca Raton, Florida, 1999, pp 29–46.

Chunlei Liu, Ye Ge, Jennifer Hou, Mike Fitz, Wei-Peng Chen, and Raj Jain. OSU-MAC: A New, Real-Time Medium Access Control Protocol for Wireless WANs with Asymmetric

Wireless Links, *IEEE International Conference on Distributed Computing Systems (ICDCS 2001)*, April 16-19, 2001, Phoenix, Arizona, USA, pp 537-546.

Chunlei Liu and Raj Jain, Delivering Faster Congestion Feedback with the Mark-Front Strategy, *International Conference on Communication Technologies (ICCT 2000)*, Beijing, China, August 21-25, 2000, Vol. 1, pp 665-672.

Chunlei Liu, Sohail Munir, Raj Jain, Sudhir Dixit, Packing Density of Voice Trunking using AAL2, *Conference Record / IEEE Global Telecommunications Conference*, v 1 (B) 1999, pp 611-615.

Arjan Durresi, Mukundan Sridharan, Chunlei Liu, Mukul Goyal and Raj Jain, Multilevel Explicit Congestion Notification, *IEEE International Conference on Computer Communications and Networks (ICCCN2001),* Scottsdale, Arizona USA, October 15 – 17, 2001.

Arjan Durresi, Mukundan Sridharan, Chunlei Liu and Raj Jain, Improving Congestion Control in Satellite Networks, *Proceedings of SPIE ITCOMM2001 Conference on Quality of Service over the Net Generation Data Networks*, Denver, August 20-24, 2001, pp 293-303.

Arjan Durresi, Mukundan Sridharan, Chunlei Liu, Mukul Goyal and Raj Jain, Multilevel Early Congestion Notification, *Proceedings of the 5th World Multiconference on Systemics, Cybernetics and Informatics SCI'2001, ABR over the Internet*, Orlando, FL, July 22-25, 2001, pp 12-17.

Mukul Goyal, Arjan Durresi, Padmini Misra, Chunlei Liu and Raj Jain, Effect of Number of Drop Precedences in Assured Forwarding,, *Conference Record / IEEE Global Telecommunications Conference*, v 1 (A) 1999, pp188-193.

FIELDS OF STUDY

Major Field: Computer and Information Science

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

Figure **Page**

CHAPTER 1

INTRODUCTION

1.1 Motivation

In the past decade, there has been a tremendous growth of wireless networks, which has been driven mainly by two factors. One is the need to access the Internet "any time, any where" [35], and the other is the massive use of mobile telephony for communications.

Today, networking has permeated every aspect of our life. The Internet has become the most convenient, efficient and cost effective way to search and exchange information. Being able to connect to customers and friends, and being able to access the web any time, anywhere has become a necessity in our businesses and day-to-day lives. Traditionally, the Internet has been a combination of fixed local area networks connected by all types of cables. Wireless technology provides a way to liberate mobile users from all cables while connecting to the information source. One-way and two-way pagers, cellular phones, wireless data modems and multimedia mobile phones are changing the way we communicate and do business.

While wireless networks provide mobility and convenience to users, the quality of service and efficiency of today's wireless networks are still far from satisfactory. Conversations may be cut off when mobile subscribers travel between cells. Wireless data connections have high bit error rates, low bandwidth and long delays. Many wireless local area networks achieve only a small portion of the advertised peak bandwidth.

In order to understand the origin of these problems, we need to know the differences between wired and wireless media.

Wireless communications come in many forms and span a large range of throughput and distances. The world of wireless data includes fixed microwave links, wireless LANs, satellite links, digital dispatch networks, one-way and two-way paging networks, diffused infrared, laser-based communications, keyless car entry, the Global Positioning System and more. In addition to its wide range of forms, wireless media is essentially different from traditional wired media in the following ways.

First, the available bandwidth of the wireless media is limited and shared by many users, and overall throughput is therefore much lower than that of wired media. A typical category 5 unshielded twisted-pair (UTP) cable can deliver 1 Gbps bandwidth in Gigabit networks, but a typical wireless WAN in GSM is designed for 9600 bps per user. In wired networks, laying more cables yields more bandwidth, so the bandwidth of wired media can be expanded infinitely. On the other hand, the bandwidth of wireless media is limited by the available radio spectrum. The allocation of the spectrum is normally controlled by government agencies. While radio spectrum experts are looking for ways to encode more bits in the spectrum, the available spectrum remains limited. In this sense, the wireless bandwidth cannot be expanded infinitely.

Second, wireless media is shared. Competition for the shared medium causes access contention. If not coordinated properly, the access contention can result in low efficiency and unpredictable channel access delay.

Third, wireless media is prone to transmission errors. Variable conditions in the natural environment, multipath reflection, noise, channel interference and fading affect the signal transmission and lead to high bit error rates. Forward Error Correction (FEC) and Automatic Retransmission Request (ARQ) can be used to recover from transmission errors. In wireless WAN systems, the average bit error rate after FEC can be as high as 10^{-3}. Link layer ARQ can reduce the bit error rate perceived by the high protocol layers to a magnitude of 10^{-7} [47]. The remaining transmission errors are usually perceived as packet losses by upper protocol layers.

Finally, wireless media is asymmetric. In the cellular phone system, the radio connection consists of a downlink from the base station to the mobile hosts, and an uplink from the mobile hosts to the base station. The two links are not symmetric. The base station is not mobile and is normally equipped with a transceiver of higher power, so the downlink usually has a lower bit error rate and a higher bandwidth. The uplink on the other hand, is shared by mobile stations in the cell. The transmission is from smaller transceivers in the handset and therefore has a higher bit error rate and a lower bandwidth.

Some of the low QoS problems of wireless networks are due to the limitations of the wireless media — wireless networks cannot achieve the same level of bandwidth and quality of service as wired networks. However, many of these problems are the result of incorrect underlying assumptions that were made based on wired networks that do not hold true in wireless networks. Most existing networking protocols were designed when the networks were composed solely of wired links. These protocols made assumptions about the

characteristics and operation of the underlying media. While these assumptions improved efficiency in wired networks, they do not apply to wireless links. Continuing to use these assumptions in heterogeneous networks that consist of wired and wireless links causes poor performance. With the rapid development of wireless networks, identifying the causes of poor performance and enhancing the current protocols has become an urgent task.

The goal of this dissertation is to address problems in current networking protocols that cause poor performance on wireless links, to revisit enhancement proposals in the literature and to point out new direction for enhancements.

1.2 Architecture of Wireless Networks

The architecture of a typical wireless network is shown in Figure 1.1. In this figure, a mobile host is communicating with a host in a fixed network connected to the Internet. The key components in this architecture include the mobile host, the base station, the switching center and the Internet. The mobile host can be a mobile phone handset or a computer with a wireless modem. The base station is a stationary collection of hardware components that communicate with nearby mobile hosts through radio media. The geographical area covered by a base station is called a cell. A cellular system typically consists of thousands of cells, each with its own base station. Together these cells cover a large region, such as a metropolitan area or an entire country.

All communication with mobile hosts is through the radio channel between the mobile host and the base station. The communication path consists of a last-hop wireless link and a number of wired links in the fixed subnetworks. The radio channel consists of a downlink and an uplink. The downlink is broadcast from the base station to all mobile hosts in the cell. The access to the uplink is normally shared by all mobile hosts. The base station is

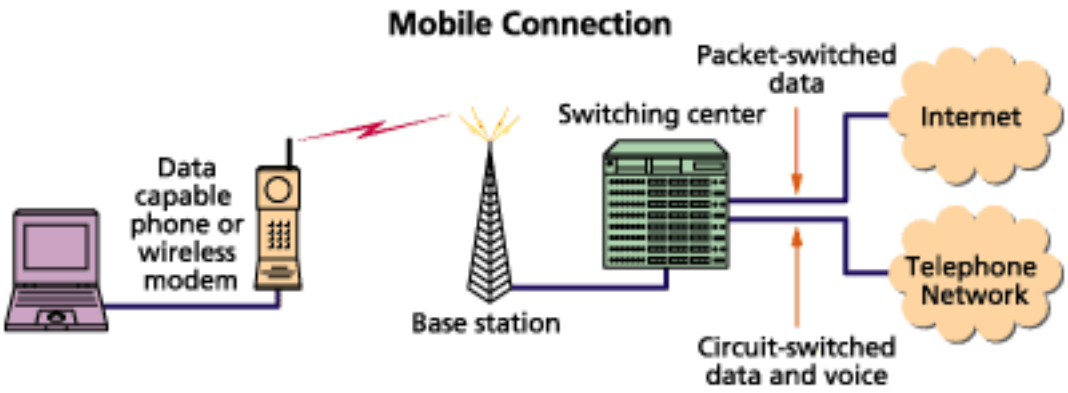


Figure 1.1: Architecture of a wireless network

responsible for scheduling the access to the media and for resolving contentions using a media access control protocol.

Transmission on wireless links is prone to higher ratio of corruption than on wired links. Data corruption causes problems in the data link layer, as well as upper layers. In the data link layer, two types of error control mechanisms, Forward Error Correction (FEC) and Automatic Retransmission Request (ARQ), can be used to combat transmission errors. In FEC, the sender of a packet calculates an error correction code according to a predetermined procedure. The error correction code is appended to the original packet and transmitted over the wireless media. When the packet is received at the other end of the data link, the receiver uses the error correction code to check whether any bits in the packet have been corrupted. Some error correction schemes have the ability to correct bit errors up to a certain level. This error control mechanism corrects errors without adding traffic in the reverse direction and is, therefore, called forward error correction. FEC may not catch all transmission errors. The remaining errors can be recovered by ARQ. Using ARQ, packets

transmitted on unreliable links are acknowledged upon receipt. If the acknowledgment of a packet is not received before a timer expires or if the packet is negatively acknowledged, the packet will be retransmitted. Retransmissions may cause variable delay and unexpected timeouts in upper protocol layers. This will be addressed later in this dissertation.

The base station is connected to a switching center through land lines. A switching center normally controls all base stations in a metropolitan area. Equipped with more intelligent components, the switching center is capable of performing complex functionalities. One function the switching center performs is voice trunking. Voice packets from different mobile terminals that share the same trunk can be packed and sent in an ATM cell to reduce packing delay.

Data packets sent to the communicating party need to go through the switching center and a number of routers before reaching their destination. In wired networks, data are transferred at much higher speed and much lower bit error rates. Packets may get lost, but the majority of packet losses are caused by congestion. In traditional tail drop routers, when the router buffer is full, the incoming packet is dropped. In TCP/IP, packet losses are regarded as a signal to notify the source of the congestion in the network. Network routers can also send explicit congestion signals by setting a bit in the IP packet header. When the packet reaches the destination, the feedback will be reflected back to the source to trigger congestion control actions.

The "last-hop" configuration shown in Figure 1.1 is very typical. The most common scenario is a mobile host retrieving information from a fixed host in the Internet. In this scenario, the majority of traffic is from the fixed host to the mobile host. Most wireless network enhancement proposals are based on this configuration. A less common scenario

is the “lost-hop” configuration with the mobile host acting as a web server. In this scenario, the majority of traffic is from the mobile host to the wired network.

In addition to this configuration, wireless links can appear in many other configurations including:

- **Satellite Links**: Satellite links are a special case in which the wireless link is in the middle of the connection. Satellite links normally have high bandwidth, large round-trip time and asymmetric paths. Performance analysis and enhancement of satellite links have been a hot research topic.

- **Mobile Networks**: Networks in moving vehicles, airplanes and ships are mobile networks. All hosts in the mobile networks are connected to other networks through an intermediate wireless link.

- **Multiple Wireless Links**: Multiple wireless links can be contained in a communication path. This is very typical for networks using microwave links in remote areas where laying wires is difficult. Another example is networks that connect isolated islands using radio links. Normally connections in such networks have more than one wireless link.

- **Ad Hoc Networks**: In ad hoc networks, participating hosts are connected by wireless links. These hosts are part of the network only for the duration of the communication session. Because of the unreliability of the hosts and links, the structure of the network can change dynamically.

All these configurations have common performance problems and need enhancement.

1.3 Challenges

The goal of this dissertation is to identify unsuitable assumptions made for the wired-only networks and to solve problems in current networking protocols that cause poor performance on wireless links. Specifically, we focus on the following challenges:

- **TCP enhancement at the mobile host**

  When wireless links are added to the network, severe performance degradation is caused by false interpretation of packet losses. Congestion control mechanisms in traditional TCP algorithm regard all packet losses as the indication of network congestion, and reduce the effective transmission rate upon detection of packet losses. The presence of transmission errors in the wireless link causes these congestion control mechanisms to make incorrect judgments about network status, take wrong congestion control actions, and cause severe performance degradation. While many enhancement proposals have been suggested in the literature, our review shows that these proposals make modifications in places that are essentially impossible. Many of them only apply to the receiver scenario of the "last-hop" configuration, and the majority of these proposals do not improve TCP's timeout behavior, which we found to be the leading contributor of performance degradation.

  The challenge is to find a wireless TCP enhancement scheme that makes modifications within the scope of wireless service providers, applies to all wireless configurations, and avoids the performance degradation completely.

- **Faster congestion feedback from network routers**

  Timely congestion feedback from the network is crucial to the effectiveness of congestion control actions taken by the TCP. Faster congestion feedback reduces the

number of packet losses during a congestion episode and the delay experienced by packets passing the congested router. Faster congestion feedback also reduces hardware cost of the routers by reducing the buffer size requirement. Historically, timeout, triple duplicate acknowledgments and Explicit Congestion Notification have been used as the mechanisms to deliver congestion feedback from the routers. Each delivers a faster feedback to the source and yields higher throughput. It has been observed that these mechanisms have some bad effects such as lock-out phenomenon [24] and unfairness.

The challenge is to find a mechanism to deliver even faster congestion feedback and yield better performance.

- **Real-Time QoS support in MAC protocol**

  As real-time applications gain more popularity and are increasingly deployed in mobile handsets, more support for real-time quality of service support is needed from the media access control protocol. The OSU wireless radio modem project has a working testbed designed to support both real-time and non-real-time applications. The testbed has two physical layer characteristics typical in many cellular wireless networks, asymmetric forward/reverse channels and half duplex transmission. A survey of current MAC protocols did not find any MAC protocols that support real-time applications on such physical radio media.

  The challenge is to design a MAC protocol that supports real-time applications while maximally utilizing the bandwidth resources in an asymmetric and half-duplex environment. Although the MAC protocol is designed for the specific physical media and the real-time application supported is light-weight, the principles and techniques

of this design can be generalized to high-speed wireless networks and to support large-scale real-time applications.

- **Quality and efficiency analysis for voice trunking**

  Voice is the most traditional real-time application, and is one of the two prominent motivations driving the tremendous growth of wireless networks. In fact, voice comprises more than 90% of today's wireless traffic. With the development of high-speed data networks, traditional circuit voice technology is giving way to the more cost-effective and more scalable packet voice protocols. The major QoS concern of packet voice is the delay associated with voice packets. Packing delay, which is the delay to pack an ATM cell with voice, becomes the largest component of the end-to-end delay when efficient voice compression algorithms are used. The AAL2 standard in the ATM Forum provides a way to reduce the packing delay. However, voice trunking has a tradeoff between voice quality and bandwidth efficiency. The standard did not specify how to make the tradeoff.

  The challenge of this research is to analyze the relationship between voice quality and bandwidth efficiency, and establish guidelines for making the tradeoff.

1.4 Main Contributions of this Study

The major contributions of this study are:

1. The Congestion Coherence algorithm that enhances TCP/IP over wireless links
2. The mark-front strategy that delivers faster congestion feedback for better congestion control

3. A MAC protocol for asymmetric wireless LAN for the OSU wireless radio modem testbed

4. A stochastic analysis to balance the voice quality and bandwidth efficiency in AAL2 voice trunking[1].

1.5 Dissertation Outline

The rest of this dissertation is organized as follows. Chapter 2 is a background introduction. We describe the TCP/IP protocol and its congestion control mechanisms. The state of the art of current research topics is also reviewed in this chapter. Chapter 3 identifies problems in the current wireless networks and states the methodology and approaches used in the research.

Chapters 4 through 9 are the core of the dissertation. The requirements on wireless TCP enhancements are stated in Chapter 4. We not only state these requirements, but also analyze why these requirements are important for enhancement proposals. These requirements serve as a set of criteria to evaluate the enhancement proposals. They establish the enhancement solution we propose in this dissertation as a practical and effective enhancement.

The leading contributors of wireless TCP performance degradation are analyzed in Chapter 5. A simulation is set up to collect performance related data for wireless TCP and major enhancement proposals. A statistical analysis shows that, contrary to common understanding, the dominant contributor of TCP performance degradation is not the inability to distinguish wireless corruption from congestion losses, but instead the timeout caused

[1]The analysis and result are adopted in the 3GPP RAN Technical Specification Group standard: Delay Budget within the Access Stratum [1].

by multiple losses in one window is the leading contributor. This analysis sets the tone for our enhancement direction.

The proposed enhancement solution called Congestion Coherence is described in Chapter 6. Analysis and simulations showing the benefits of this proposed solution and comparison with other enhancement proposals are presented.

Efficient TCP algorithms needs fast congestion feedback from the network. A new mechanism is suggested in Chapter 7 to deliver faster congestion feedback than current mechanisms, i.e., timeout, triple duplicate acknowledgments and Explicit Congestion Notification.

Chapter 8 is devoted to a Media Access Control protocol designed specifically for the narrow-band wireless radio modem testbed in the Ohio State University Department of Electric Engineering and Department of Computer and Information Science. This protocol is a unique protocol focusing on providing real-time support in an asymmetric wireless local area network. Although it is designed for a specific system, the principles used in this protocol can be generalized to high-speed wireless networks to support both real-time and non-real-time applications, and to improve wireless bandwidth efficiency in asymmetric wireless LANs.

Chapter 9 analyzes the packing process in AAL2 voice trunking. A Markovian model is established to analyze the relationship of packing timer expiration value and packing density. This analysis provides a guideline on the tradeoff between voice quality and bandwidth efficiency.

Finally, in Chapter 10, we summarize the dissertation and point out the open issues for future study.

CHAPTER 2

BACKGROUND AND STATE OF THE ART

This chapter serves as an introduction to the background material and the state of art. The background introduction includes an overview of TCP/IP, an anatomy of TCP congestion control mechanisms and a comparison of different TCP variants. The state of the art consists of surveys on wireless TCP enhancements, network congestion feedback mechanisms and media access control protocols.

2.1 Transmission Control Protocol Overview

Transmission Control Protocol (TCP) [63, 12, 2] is the most widely used transport protocol. It was designed to be connection-oriented on an unreliable network layer. By providing reliable data delivery and congestion control on the top of the Internet Protocol (IP), TCP forms the foundation of today's Internet. The operation of the TCP algorithm is a concerted effort of the sender, the network and the receiver to reliably deliver user data across an unreliable network or a set of inter-connected networks. This section provides an overview of the TCP algorithm. TCP's congestion control mechanisms and congestion feedback mechanisms from the network are discussed in following sections.

IP is not designed to recover from packet losses, nor does it guarantee traffic delivery. Packets sent on IP networks may be lost when congestion or transmission errors happen, or

when the number of permissible transit hops is exceeded. In addition, IP does not deliver packets in order. At the destination, the IP layer forwards packets to the upper layer even though they may be out of order.

Certain applications require assurance that datagrams have been delivered safely to the destination. Lost and corrupted packets need to be retransmitted. Out-of-order packets need to be rearranged before delivery to the application. Some applications use the receipt acknowledgments from the destination to control the sending process. The mechanisms to achieve these important services are built in the TCP algorithm.

TCP belongs to the transport layer of the conventional seven-layer model, sitting above IP layer and below upper layers. The TCP algorithm resides in the host computer or a machine tasked with the end-to-end integrity of the transfer of user data. It is not loaded into intermediate routers to support user data transfer. If TCP runs in the router, it does so to support activities such as network management, terminal sessions with the router, etc.

2.1.1 Establishment and Termination of TCP Connections

TCP is a connection oriented service. Before sending any user data, a connection must be established between the source and the destination. To open a connection, the sender transmits a SYN segment, the receiver replies with its own SYN, and then the sender answers with a SYN-ACK segment. The SYN segments can also carry control bits to negotiate options used in the TCP algorithm. After the connection is established, data can be transmitted in both directions. This procedure of establishing the connection is called a *three-way handshake*.

After all the data is sent, the sender and the receiver exchange FIN and FIN-ACK segments to terminate the connection.

2.1.2 Acknowledgments and Delayed Acknowledgments

TCP achieves reliable transport by requiring positive acknowledgment for received segments. Each byte of the data is assigned a unique sequence number and the receiver sends an acknowledgment (ACK) upon receipt of a segment. TCP acknowledgments are cumulative — an ACK confirms all bytes up to the given sequence number.

Acknowledgments are sent only in response to a packet being received, rather than at regular intervals. Thus if no packets are arriving, no acknowledgments will be sent. Cumulative acknowledgments are efficient in the sense that an ACK does not have to be sent for every packet received. However, they are ambiguous because they do not explicitly inform the sender of any packets which have been lost or damaged. Beyond the acknowledged bytes, the sender has no information whether all data has been received.

TCP acknowledgments can be a separate packet, or can be "piggy backed" onto a data packet in the reverse direction. Normally TCP does not acknowledge a received segment immediately, but waits for a certain period. If a data segment is sent during this period, the acknowledgment is "piggy backed" onto it. Alternatively, the receiver can wait until another data segment arrives, and acknowledge both received segments at once. However, TCP must not delay acknowledgments for more than half a second and should send an acknowledgment for every other received segment [2].

TCP acknowledgments also drive TCP's transmission. In the equilibrium condition, each arriving ACK triggers a transmission of a new segment. This is called TCP's *self-clocking* property.

2.2 TCP Congestion Control Mechanisms

Congestion occurs when the total amount of data pumped into the network exceeds the network's capacity. During congestion the queue at the congested router grows larger so packets going through the router suffer an extended delay. As congestion continues, the queue size may exceed the buffer size and cause buffer overflow. Packets that cannot be accommodated in the buffer are dropped by the congested router, and have to be retransmitted later from the source. In many cases when the dropped packet is retransmitted, some packets that have already passed the congested router are also retransmitted. Retransmissions use precious bandwidth and add a variable delay to the retransmitted packets. Congestion leads to severe throughput degradation and a dramatic increase in the packet transfer delay and is therefore very undesirable.

Figure 2.1 from Chiu and Jain [18] illustrates the delay and throughput of a network in response to increasing load. When the load is low, an increase in load improves the throughput without significantly raising the delay. When the load comes close to the network capacity, further increases in load raise the delay but not the throughput. The point in the figure when the load equals the network capacity is called the *knee*, and the point when the throughput falls is called the *cliff*.

Finding the network's capacity is not an easy task. TCP's strategy is to control congestion once it happens, as opposed to trying to avoid congestion in the first place. In fact, TCP repeatedly increases the load on the network in an effort to find the point at which congestion occurs, and then backs off from this point. In other words, TCP needs to create losses to find the available bandwidth of the connection. This type of strategy is called *congestion control*. An alternative, which is called *congestion avoidance*, is to predict when

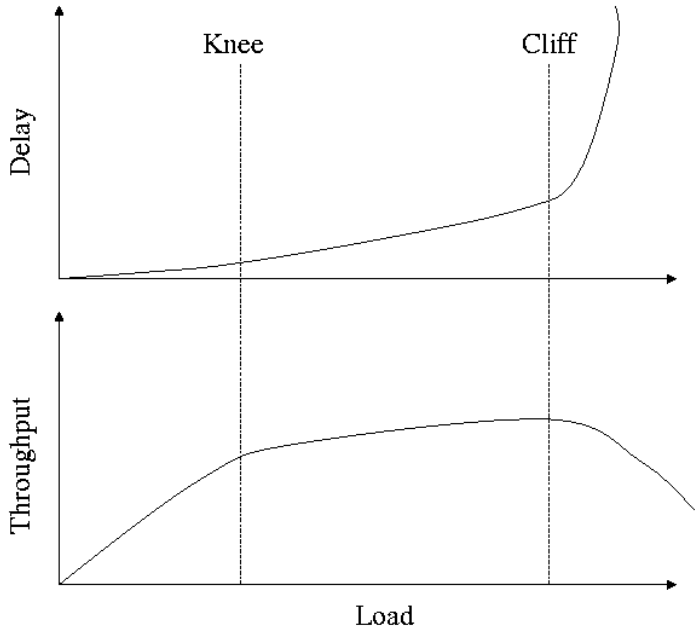


Figure 2.1: Load versus delay and throughput

congestion is about to happen and then reduce the transmission rate just before packets start being discarded.

The goal of congestion avoidance is to maintain operation at the knee, where the delay is low and the throughput is high, while the goal of congestion control is to maintain operation to the left of the cliff. TCP congestion control contains mechanisms that enable the sender to quickly estimate the network capacity, reach steady-state operation, detect and react to network congestion, and recover from congestion losses. These mechanisms are described below.

2.2.1 The AIMD Principle

Both congestion control and congestion avoidance problems can be formulated as system control problems in which the system detects its state and feeds this back to its users who adjust their controls. The key component of congestion control and congestion avoidance schemes is the control function used by the users to increase or decrease their load. Chiu and Jain [18] abstractly characterized these functions as a number of increase and decrease algorithms. They proved that a simple Additive Increase and Multiplicative Decrease (AIMD) algorithm satisfies the sufficient conditions for convergence to an efficient and fair state regardless of the starting state of the network.

Optimization considerations require that the change in efficiency and fairness be maximized in every feedback cycle. Using these considerations, AIMD was shown to be the optimal policy.

2.2.2 Slow Start and Congestion Avoidance

TCP is designed to be conservative in the amount of data transmitted to the network to prevent congestion. The mechanisms to estimate network capacity are *slow start* and *congestion avoidance*, which were first introduced by Jacobson [39] and were quickly made mandatory in RFC 1122 [12].

Slow start and congestion avoidance are two phases in TCP transmission. These two phases are differentiated by two variables: congestion window (*cwnd*) and slow start threshold (*ssthresh*). *cwnd* is a sender-side limit on the amount of data the sender can transmit into the network before receiving an ACK. The receiver's advertised window (*rwnd*) is a receiver-side limit on the amount of outstanding data. The minimum of *cwnd* and *rwnd* governs data transmission [2]. *ssthresh* is needed to determine whether the transmission is

in the slow-start or the congestion avoidance phase. If *cwnd* is smaller than *ssthresh*, the transmission is in the slow-start phase. Otherwise, the transmission is in the congestion avoidance phase.

The initial value of *cwnd*, called *initial window* (IW), must be no more than two segments. After the connection is established, the TCP sender transmits as many segments to the network as the value of IW permits. Before receiving an ACK, the sender is blocked and cannot send any new data. After receiving the ACK, the number of new segments to be sent is equal to the number of acknowledged segments plus one segment due to increase of *cwnd*.

During slow start, a TCP sender starts with a *cwnd* of one or two segments and increase the *cwnd* by at most sender maximum segment size (SMSS) bytes for each ACK received that acknowledges new data. If no packet is lost, the congestion window is one segment in the first RTT, two segments in the second RTT, and three segments in the third RTT. Slow start is somewhat of a misnomer because the growth of *cwnd* is actually very fast. It is called slow start because the transmission starts from a small value, normally one segment.

The exponential growth in slow start ends when *cwnd* reaches *ssthresh* or when congestion is detected. The sender can detect congestion in two ways: the expiration of the retransmission timer or the receipt of three consecutive duplicate acknowledgments (dupack). The data receiver sends a dupack after receiving an out-of-order segment. When the third dupack has arrived, the TCP sender goes to a fast retransmit/fast recovery algorithm.

During congestion avoidance, *cwnd* is incremented by one full-sized segment per round trip time (RTT). That means that the *cwnd* is increased only after a fill window of data is acknowledged. Congestion avoidance continues till the end of the connection or until

congestion is detected by RTO. A commonly used formula to update *cwnd* is [2]

$$cwnd = cwnd + \text{SMSS} * \text{SMSS}/cwnd \tag{2.1}$$

2.2.3 Timeout

Traditionally a TCP receiver only acknowledges the highest segment it has successfully received in order. In case of a segment loss, the receiver cannot acknowledge new data, so the sender is not allowed to send new data.

Therefore, before the retransmission timeout, there is usually an idle period during which the sender does not transmit any data, possibly causing the communication path to be under-utilized. There is no way to indicate the loss to the sender.

In early versions of TCP, the only way to recover from a segment loss was to wait for the *retransmission timeout* (RTO) timer to expire. The value of RTO is determined from the measured *round-trip time* (RTT) and the variance of recent RTT measurements [39, 61].

Timeout is a mechanism to battle congestion, but the timeout timer must be set correctly. If the timer is too long, the network may be idle for a long time before the lost packet is retransmitted. If the timer is set too short, the sender may unnecessarily retransmit a segment, which only adds to network congestion.

Another reason for needing an accurate timeout value is that timeouts are regarded as an implication of congestion and have significant impact on congestion control algorithm. Inaccurate timeout values may trigger false congestion control actions resulting in low throughput.

2.2.4 Round-Trip Time Measurement

Estimating round-trip time is not as easy as it sounds. An intuitive idea is to record the time when TCP sends out a packet, and read the time again when the acknowledgment is

received. The difference between the two times is the round-trip time. Running average can be used to smooth the variations of individual samples:

$$\text{EstimatedRTT} = \alpha \times \text{EstimatedRTT} + (1 - \alpha) \times \text{SampleRTT} \tag{2.2}$$

Here α is an averaging parameter between 0 and 1. The timeout value is set to a quite conservative value:

$$\text{Timeout} = 2 \times \text{EstimatedRTT} \tag{2.3}$$

After several years of use in the Internet, a flaw of the simple algorithm was discovered. The problem is that data packets and acknowledgments can be lost in the network. When a packet is retransmitted and the acknowledgment arrives at the sender, it is impossible to determine if this acknowledgment should be associated with the first or the second transmission.

Karn and Patridge [44, 45] suggested a simple solution which only measures sample RTT for segments that have been sent once. If a packet is retransmitted, TCP stops taking samples of the RTT for the retransmitted packet. They also propose to change TCP's timeout mechanism. The timeout value doubles each time the lost packet is retransmitted, similar to the exponential backoff in Ethernet.

A timestamp option was recommended in RFC 1323 [40]. When using the timestamp option, the TCP sender writes the current value of a "timestamp clock" into a 12-byte optional field in the header of each outgoing segment. The receiver then echos those timestamps in the corresponding ACKs. When receiving an ACK, The sender can determine the RTT by calculating the difference between the current value of its "timestamp clock" and the timestamp echoed in the ACK.

A few years later, Jacobson and Karels proposed more dramatic changes to TCP to fight congestion, including a new method to set the timeout value. In this new method, the timeout value is set according to the estimated RTT as well as the deviation of individual RTT samples [39]:

$$\text{Difference} = \text{SampleRTT} - \text{EstimatedRTT} \tag{2.4}$$

$$\text{EstimatedRTT} = \text{EstimatedRTT} + \delta \times \text{Difference} \tag{2.5}$$

$$\text{Deviation} = \text{Deviation} + \rho \times (|\text{Difference}| - \text{Deviation}) \tag{2.6}$$

The timeout value is set to

$$\text{Timeout} = \text{EstimatedRTT} + \eta \times \text{Deviation} \tag{2.7}$$

Here, δ and ρ are constants between 0 and 1. Paxson [61] states that δ should be set to 1/8, ρ to 1/4 and η to 4. These values are widely used in present TCP implementations because of their efficiency in binary arithmetics. The above algorithm was introduced in [39], because the original algorithm introduced in [63] turned out to be inadequate with congested internetworks. This algorithm was made mandatory in RFC 1122 [12] by stating that TCP implementations must follow the rules mentioned above. The retransmission timer must be reset when an ACK is received that acknowledges new data. This way the timer will expire after RTO seconds. For a more detailed description, refer to [61].

2.2.5 Fast Retransmit and Recovery

Timeout is a mechanism to deliver the packet loss information and trigger the retransmission from the sender, but timeouts are always accompanied by an extended period during which no data can be sent. A better mechanism for passing negative acknowledgments and triggering retransmissions called *triple duplicate acknowledgments* is introduced in

RFC 2001 [70]. This mechanism regards the receipt of three consecutive duplicate acknowledgments as an indication of packet loss. Duplicate acknowledgments acknowledge exactly the same octets as previous acknowledgments. The receiver generates a duplicate ACK when it receives an out-of-order segment. The algorithm of triggering a retransmission on three duplicate ACKs is called *fast retransmit*.

Additionally, the congestion control specification also improves the TCP performance by allowing the sender to retransmit a new segment each time an additional duplicate ACK arrives after the fast retransmit if the TCP congestion window and receiver window permit. This is based on the judgment that each duplicate ACK is an indication of a packet that has arrived at the receiver, meaning that the network load was reduced by one packet. This is called the *packet conservation rule* [39], which requires that the number of outstanding segments in the network is maintained during the *fast recovery* phase. However, because the packet loss can be a notification of congestion, the sender has to wait for the number of outstanding segments to be halved before new segments can be transmitted. The algorithm following the fast retransmit is called *fast recovery*, and it is completed after the first acknowledgment for new data arrives to the sender.

2.3 TCP Implementations and Extensions

TCP has been a rather fluid protocol over the last several years, especially in its congestion control mechanism. In fact, TCP is no longer defined by a specification, but rather by implementations. Due to the availability of intermediate versions of the code and the fact that patch has been layered on top of patches, one could not find a universal agreement about which technique was introduced in which release. In today's Internet, many different versions of TCP implementation coexist and communicate with each other.

In this section, we review the different TCP implementations and extensions. The purpose of this review is to trace the development of the TCP protocol and to understand the TCP versions that new TCP enhancements must talk to.

2.3.1 TCP Tahoe

TCP Tahoe, which is also known as BSD Network Release 1.0 (BNR1), refers to the original implementation of Jacobson's congestion control mechanisms on the BSD operating system in 1988. It includes slow start, congestion avoidance, an improved round-trip-time estimation and fast retransmit [39]. The new mechanisms were introduced in response to congestive collapses that began occurring on the Internet in 1986 and caused throughput to drop in some cases by a factor of a thousand. TCP Tahoe includes most modern TCP congestion control mechanisms and is the predecessor of most other TCP implementations.

The goal of these mechanisms is to ensure that a TCP connection is able to reach a state of equilibrium and that the connection obeys the "conservation of packets" principle once it is in equilibrium. This principle states that once a connection has reached equilibrium, it should only transmit a packet on the network when it receives feedback indicating that a packet has left the network. The connection reaches equilibrium by probing the network for available bandwidth and adjusting a newly proposed sender congestion window. In TCP Tahoe, the window used by the sender is taken as the minimum of the receiver window and this new congestion window.

The most significant performance improvement of TCP Tahoe comes from the fast retransmit mechanism. By triggering a retransmission upon the receipt of three duplicate ACKs, TCP Tahoe avoids the long wait for the retransmission timer to expire.

The main disadvantage of TCP Tahoe is its frequent initiation of the slow start following each packet loss. The slow-start process takes a long time to bring the TCP transmission rate back to normal, so TCP Tahoe does not perform well on high bandwidth-delay product connections.

2.3.2 TCP Reno

TCP Reno was implemented in the BSD operating system in 1990, and is also known as BSD Network Release 2.0 (BNR2). It is the most widely used TCP version today. On top of TCP Tahoe, TCP Reno adds the fast recovery mechanism. Fast recovery enables the connection to quickly recover from isolated segment losses [70].

The main difference between TCP Tahoe and Reno is the phase they enter after a fast retransmit. In both Tahoe and Reno, fast retransmit is triggered upon the receipt of three duplicate ACKs. However, when a partial or new ACK is received, TCP Tahoe sets its *cwnd* to one segment. As a result, TCP enters the slow start phase until *cwnd* reaches *ssthresh*. But in TCP Reno, when a partial or new ACK is received, *cwnd* is set to *ssthresh*, causing the system to immediately enter the congestion avoidance phase.

Besides fast recovery, TCP Reno also supports *header prediction* and *delayed acknowledgments* options. Header prediction is an optimization for the common case that segments arrive in order, and the delayed acknowledgments option acknowledges every other segment rather than every segment.

The key advantage of TCP Reno over TCP Tahoe is that it maintains the clocking of new data with the duplicate ACKs and avoids initiating the slow start phase when the TCP transmission rate is reduced. This improvement is most noticeable on large bandwidth-delay product connections where the slow start phase is long.

2.3.3 TCP New Reno

The fast retransmit and recovery in TCP Reno are effective in recovering from isolated packet losses. However, it was observed that Reno does not work well if multiple packets are dropped during a single round trip [36]. In high-speed links, a typical congestion episode can cause multiple segments to be dropped. In this case, fast retransmit and recovery are not able to recover from the losses and slow start is triggered frequently. Figure 2.2 shows a case when three consecutive packets are lost from a window and the sender TCP incurs fast retransmit twice and then times out. At this time, *ssthresh* is set to one-eighth of the original congestion window value. As a result, the exponential growth lasts a very short time, and the linear increase begins at a very small window. Thus, TCP transmits at a very low rate and loses much throughput.

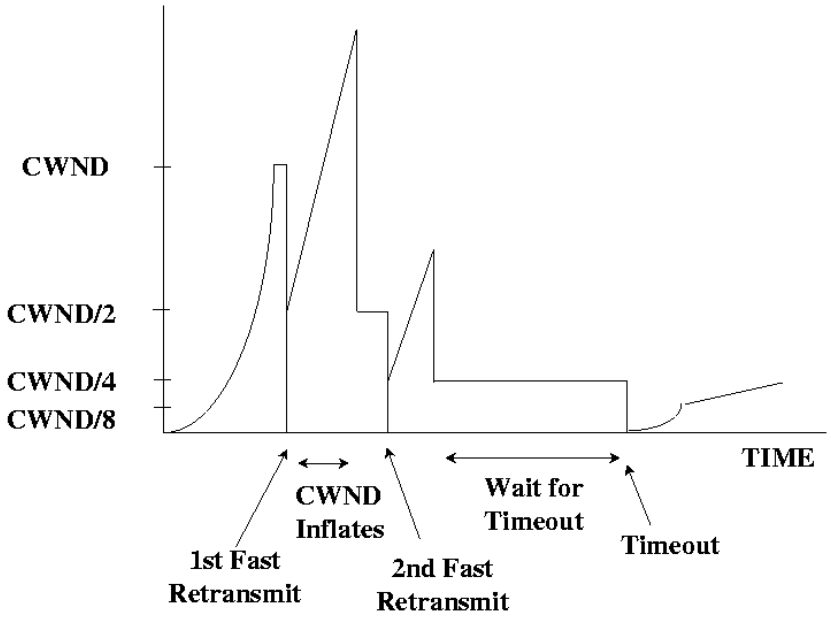


Figure 2.2: Fast Retransmit and Recovery

TCP New Reno [27] introduced a *fast-recovery phase* during which the sender remembers the highest sequence number sent (RECOVER) when the fast retransmit is first triggered. After the first unacknowledged packet is retransmitted upon the receipt of three duplicate ACKs, the sender follows the usual fast recovery algorithm and increases the *cwnd* by one for each duplicate ACK it receives. When the sender receives an acknowledgment for the retransmitted packet, it checks if the ACK acknowledges all segments including RECOVER. If so, the ACK is a new ACK. The sender exits the fast retransmit-recovery phase and sets its *cwnd* to *ssthresh* and starts a linear increase (congestion avoidance).

On the other hand, if the ACK is a partial ACK, i.e., it acknowledges the retransmitted segment and only a part of the segments before RECOVER, then the sender immediately retransmits the next expected segment as indicated by the ACK. This continues until all segments including RECOVER are acknowledged.

This modification allows TCP New Reno to handle multiple losses with a single fast recovery phase, while TCP Reno has to invoke fast-recovery multiple times. Avoiding multiple fast-recovery phases can eliminate two detrimental effects. First, it prevents the sender from shrinking the congestion window too dramatically. Second, it eliminates stalls that often happen to TCP Reno after two packets are dropped in a single window.

While TCP New Reno is capable of handling multiple packet losses in a single window, it is limited to detecting and resending at most one packet per round-trip-time. This deficiency becomes more pronounced as the delay-bandwidth product becomes greater.

2.3.4 TCP SACK

Currently a TCP receiver only acknowledges the highest segment it has successfully received in order. In the event of packet loss, the sender can only tell the first lost segment

from the acknowledgments it receives. The delivery status of subsequent segments will not be known until the acknowledgment of the first retransmitted packet is received. Therefore, multiple losses within one round-trip time normally cause a timeout at the source.

The TCP SACK option specified in [57] provides a way of furnishing more information about the packets that have been received. A SACK option is an optional field in the TCP header. It is sent whenever out of sequence data is received. All duplicate ACK's contain the SACK option. The option contains a list of contiguous blocks of data already received by the receiver. Each data block is identified by the sequence number of the first byte in the block (the left edge of the block) and the sequence number of the byte immediately after the last byte of the block. Because of the limit on the maximum TCP header size, at most three SACK blocks can be specified in one SACK packet.

The receiver keeps track of all out-of-order data blocks received. When a SACK is generated, the first SACK block specifies the block of data containing the most recently received data segment. This ensures that the receiver provides the most up to date information to the sender. After the first SACK block, the remaining blocks can be filled in any order, but the receiver should try to include as many distinct blocks as possible.

The sender keeps a table of all the segments sent but not ACKed. When a segment is sent, it is entered into the table. When the sender receives an ACK with the SACK option, it marks all the segments specified in the SACK option blocks as SACKed. The entries for each segment remain in the table until the segment is ACKed. When the sender receives three duplicate ACKs, it retransmits the first unacknowledged packet. During the fast retransmit phase, when the sender is sending one segment for each duplicate ACK received, it first tries to retransmit the holes in the SACK blocks before sending any new segments. When the sender retransmits a segment, it marks the segment as retransmitted in

the table. If a retransmitted segment is lost, the sender times out and performs slow start. When a timeout occurs, the sender resets the SACK table.

The sender also uses a new pipe mechanism to track the number of packets currently in transit. SACK performs fast-retransmit just like TCP New Reno. It enters the fast-retransmit phase when a loss is detected, and it exits when all of the data that was outstanding at the beginning of the fast-retransmit phase has been acknowledged.

SACK improves upon TCP New-Reno when multiple packets are lost in a single window. The detailed information provided by SACK allows the sender to retransmit multiple lost packets within a single round trip time, while New Reno has to wait for the ACK of the first retransmitted packet to determine which other packets have been lost.

2.3.5 TCP Net Reno

Another improvement on multiple losses is the TCP Net Reno [54] proposed by Lin and Kung. They observed that 85% of the timeouts in current TCP implementations on the Internet are caused by windows too small for fast retransmission to trigger. In such cases, there are not enough packets in the network pipe to trigger the fast retransmit and recovery, and therefore timeouts are incurred. They propose a network-sensitive version of Reno called Net Reno that improves performance by reducing retransmission timeouts. In Net Reno, a new packet is sent for each duplicate ACK received before fast retransmission is triggered. In case of small windows, these new packets trigger more duplicate ACKs that trigger fast retransmit.

2.3.6 TCP Vegas

TCP Vegas [14] is a big change from earlier implementations of TCP that were distributed in releases of 4.3 BSD Unix. It uses three techniques to improve TCP throughput and reduce packet loss.

First, Vegas enhances the TCP Round Trip Time estimation algorithm by using the system clock instead of the coarse granularity TCP timer. The more accurate RTT estimate is used to sense congestion so it can retransmit a packet even before the receipt of three duplicate ACKs. Certain ACKs are used to check for timeout and retransmission using the enhanced RTT estimate.

Second, Vegas monitors changes in throughput by comparing measured throughput to expected throughput calculated using the window size. The congestion avoidance algorithm uses this information to maintain the optimal amount of data in the network. Thus, the steady state estimation is continually performed using throughput as an indicator.

Finally, Vegas modifies the slow start algorithm by introducing a constant congestion window phase every other round trip. The congestion window increases exponentially every other round trip and stays constant during the interim round-trip. During the constant phase, the algorithm compares the estimated and achieved throughput. The difference of the two throughputs is used to decide whether TCP should enter the increase or the decrease mode. TCP Vegas takes longer to get to steady state, but the steady state estimation is more accurate.

2.4 Congestion Feedback Mechanisms

The TCP algorithm at the sender and the receiver is only part of the congestion control effort. An equally important part is the congestion feedback from the network, which is the basis of any congestion control actions.

Congestion feedback can be delivered in different ways. The most primitive feedback is packet drops. In case of buffer overflow, the network simply drops incoming packets that cannot be accommodated in the router buffer. The source can detect the packet drop either from a timeout or from duplicate acknowledgments. Tail Drop is the most common packet dropping policy. More sophisticated routers can choose packets to drop according to accounting information of each connection, or drop packets before the buffer overflows in order to achieve certain performance goals.

The second approach is to send congestion control messages when congestion occurs. Based on what information is sent and where the congestion control messages are sent, there are several protocols.

The third feedback approach is to mark congestion information in the header of passing packets. When the packet reaches the receiver, the attached feedback is reflected back to the sender through acknowledgment packets. This approach is called explicit notification. When the marking policy is properly tuned, this approach can achieve the goals of congestion control without losing user data and without adding extra traffic.

These congestion feedback mechanisms are reviewed and discussed in this section.

2.4.1 Tail Drop

Tail Drop is the simplest packet dropping policy. When the router buffer is full, incoming packets that cannot be accommodated are simply dropped. Beside dropping the

incoming packet, the router performs no other operations. When the source times out, it realizes that a packet is lost, and assumes the network is congested. An alternative to timeout is duplicate acknowledgments. If there are enough packets following the dropped packets, the source can deduce the loss signal from duplicate acknowledgments. In this way, the congestion signal is delivered to the source.

Despite its simplicity, Tail Drop has a number of drawbacks, including unfairness and global synchronization.

At first glance, the Tail Drop policy seems fair to all packets: no packet is treated differently from others. In fact, Tail Drop is unfair to bursty connections and new connections. Compared with smooth connections, bursty connections have greater probability to fill up the buffer space than smooth connections. If the same number of packets are sent, a bursty connection will suffer more packet drops than a smooth connection, causing unfairness against the bursty connection. Tail Drop also favors old connections than new connections. Old connections that occupy the buffer space can keep packets of new connections from entering the buffer. In some situations, Tail Drop allows a single connection or a few connections to monopolize the queue space, preventing other connections from getting into the queue. This is called the lock-out phenomenon [24].

Tail Drop is also blamed for causing global synchronization. When the buffer is not full, all packets are successfully enqueued and forwarded, causing the congestion window of all connections to grow. Thus the network is quickly congested. As the buffer gets close to full, packets of all connections have a good chance to be dropped, causing all connections to back off simultaneously. This results in a sustained period of lowered link utilization, reducing overall throughput. Even though at extremely low loads the performance of Tail Drop is acceptable, many papers have shown that Drop Tail is always outperformed by

other policies [48, 52]. Kim [46] even shows that the entire system can be shut down if Drop Tail is implemented.

For IP over ATM traffic, Tail Drop in ATM may result in some cells of a packet being dropped while other cells may be enqueued and forwarded. Floyd [30] shows that the tail drop scheme causes poor TCP performance because of phase effects in bursty TCP traffic.

2.4.2 Partial Packet Discard

Partial Packet Discard (PPD) was first proposed in [68] to efficiently transport TCP segments over ATM networks. When PPD has to drop a cell due to buffer overflow, all subsequent cells belonging to the same packet are dropped. PPD avoids transmitting cells belonging to a corrupt packet that must be retransmitted entirely by higher-layer protocol (e.g., TCP), thus improving the utilization of bandwidth resources.

2.4.3 Early Packet Discard

EPD [68] was also designed to avoid the transmission of partial packets in an ATM network. Both EPD and PPD use the End of Message (EoM) cell information to distinguish end of frame boundaries. In the PPD scheme, when a cell is dropped due to buffer overflow, the remaining cells from the frame are also dropped. EPD drops complete packets instead of partial packets. As a result, the link does not carry incomplete packets that would have been discarded during reassembly. In EPD, a threshold less than the buffer size is set in the buffer. When the switch queue length exceeds this threshold, all cells from any new packets are dropped. Packets that had been partly received before exceeding the threshold are still accepted if there is buffer space. The superior performance of EPD was demonstrated over Tail Drop and PPD [48, 52].

EPD does not discriminate between connections in dropping packets. In many cases, EPD drops packets belonging to one connection causing it to back up, but transmits packets belonging to another connection allowing it to send more packets. A number of dropping policies including Selective Drop [33], Early Packet Discard with Fair Buffer Allocation [33] and Early Selective Packet Discard [17] were proposed. These policies maintain and use per-connection accounting information to achieve better fairness among connections.

2.4.4 Drop-from-Front

The Drop-from-Front option was first introduced by Yin [76]. When the buffer is full, instead of dropping the incoming packet, the congested router makes room for the incoming packet by dropping the packet in the front of the queue. In this way, the TCP sender is notified about the congestion one entire buffer sooner than if the Tail Drop had been applied. The sender can respond to the congestion sooner and thus shorten the congestion period.

Lakshman etc. [51] studied a variant of Drop-from-Front used on IP over ATM, called Partial Frame Drop at the Front (PFDF). In PFDF, when a cell is dropped from front, the packet to which the cell belongs is marked so that all remaining cells from the same packet are also dropped when they arrive at the front. The study showed that PFDF achieves better throughput than Partial Packet Discard, Drop-from-Front and Tail Drop, especially in high bandwidth-delay product environments.

2.4.5 Random Early Detection

Random Early Detection (RED) [31] is a congestion avoidance mechanism implemented in routers to perform active queue management. RED differs from other packet dropping policies in that routers need not to maintain connection state information. It is

designed to work in collaboration with a transport layer congestion control protocol, such as the congestion control in TCP. The objectives of RED include avoiding global synchronization and keeping the average queue length in a region of low delay and high throughput. RED avoids the lock-out phenomenon and improves fairness among connections. Bursty connections also get fairer treatment than in Tail Drop. Bursty connections are more likely to cause congestion and get dropped in Tail Drop, but they do not consume more bandwidth on average.

The RED algorithm consists of two parts: the estimation of the average queue size q_{ave} and the calculation of the dropping probability for the incoming packet. Basically, RED uses an exponentially weighted moving average queue length, but the implementation has to consider idle periods during which no packets arrive and the queue length is not updated.

The probability to drop a packet is determined by a particular algorithm. The two parameters, min_{th} and max_{th}, represent the lower and upper limits of average queue length for packet dropping. With average queue length below min_{th}, no packet will be dropped. When average queue length is above max_{th}, all incoming packets will be dropped deterministically. When the average queue length is between the two thresholds, the incoming packet will be dropped with a probability calculated from q_{ave} and a configured maximum drop probability p_{max}.

Dropping packets with probability in this way ensures that TCP connections back off irregularly to break the global synchronization. The queue length at the router no longer oscillates as much as in the Tail Drop. The average queue length can be kept in low level so the router will be occupied to ensure high throughput and a low packet transfer delay.

Weighted RED [23] is a commercial implementation of RED that distinguishes packets with different priorities. In this scheme, hosts or edge routers may add precedence values to

packets as they enter the network. In the interior of the network, when the RED threshold is reached, higher priority packets are dropped with a lower probability than lower priority packets. In this way, W-RED can be used to provide priority-based quality of service to data flows.

2.4.6 DECbit

Packet marking provides a different type of congestion feedback than packet dropping. The DECbit was the first attempt to add a single binary bit in the packet header. A router sets this bit in a packet if its average queue length is greater than or equal to 1 at the time the packet arrives. This average queue length is measured over a time interval that spans the last busy idle cycle, plus the current busy cycle.

On the host side, the source records how many packets have the congestion bit set. If less than 50% of the packets in the last window have the congestion bit set, then the source increases the congestion window by one packet. If 50% or more packets in the last window have the congestion bit set, then the source decreases the congestion window to $\frac{7}{8}$ of the previous value. The "increase by 1, decrease by $\frac{7}{8}$" rule was selected because additive increase/multiplicative decrease makes the mechanism stable.

2.4.7 Explicit Congestion Notification

The Explicit Congestion Notification proposed in [26, 65] is a binary feedback for TCP/IP. Rather than dropping packets during buffer overflow, ECN provides a light-weight mechanism for routers to send a direct indication of congestion to the source. Although ECN is essentially a binary feedback scheme, it uses two bits in the IP header and two bits in the TCP header to ensure compatibility with other protocols and reliable delivery of the congestion feedback. The ECN-Capable Transport (ECT) bit is set by the data sender

to indicate that the end-points of the transport protocol are ECN-Capable. The CE (Congestion Experienced) bit is set by the router to indicate congestion to the end nodes. The ECN-Echo flag is set by the data receiver to notify the sender that the received packet has experienced congestion. The Congestion Window Reduced bit is used by the sender to inform the receiver that congestion action has been taken.

Bits 6 and 7 in the IPv4 TOS octet are designated as the ECN field. Bit 6 is designated as the ECT bit, and bit 7 is designated as the EC bit. In the TCP connection setup phase, the source and the destination exchange information about their desire and/or capability to use ECN. Subsequent to the completion of this negotiation, the TCP sender sets the ECT bit in the IP header of the data packets to indicate to the network that the transport is capable and willing to participate in ECN for this packet. If the TCP connection does not wish to use ECN notification for a particular packet, the TCP sender sets the ECT bit to 0 and the receiver ignores the CE bit in the received packet. When the router detects congestion, it checks whether the ECT bit of the incoming packet is set. If not, the packet will be discarded like a conventional router. If the ECT bit is set, the router will mark the CE bit to indicate to the data receiver that this packet has experienced congestion.

In the TCP header, bit 9 in the Reserved field of the TCP header is designated as the ECN-Echo flag, bit 8 is designated as the CWR bit. When a packet with the CE bit set reaches the receiver, the receiver responds by setting the ECN-Echo flag in the next outgoing ACK for the flow. To provide robustness against the possibility of a dropped ACK packet carrying an ECN-Echo flag, the TCP receiver must set the ECN-Echo flag in a series of ACK packets. The receiver uses the CWR flag to determine when to stop setting the ECN-Echo flag.

When an ECN-Capable TCP reduces its congestion window for any reason (because of a retransmit timeout, a Fast Retransmit, or in response to an ECN notification), TCP sets the CWR flag in the TCP header of the first data packet sent after the window reduction. If that data packet is dropped in the network, then the sending TCP will have to reduce the congestion window again and retransmit the dropped packet. Thus, the Congestion Window Reduced message is reliably delivered to the data receiver. After the receipt of the CWR packet, acknowledgments for subsequent non-CE data packets do not have the ECN-Echo flag set.

2.4.8 FECN and BECN

Forward explicit congestion notification (FECN) is a bit in the header of a frame-relay frame. It can be set by any node in a virtual path in the frame-relay network that is in danger of getting congested. It might be set in frames that are traveling on the virtual path from source to destination. The bit indicates the congestion to the destination node. This information should be passed-on to a higher level protocol, which should take appropriate measures.

Backward explicit congestion notification (BECN) is a Frame Relay message that notifies the sending device that there is congestion in the network. A BECN bit is sent in the opposite direction in which the frame is traveling, toward its transmission source.

2.4.9 Source Quench

Source quench [62] is another explicit feedback mechanism to tell a host computer that it needs to reduce the pace at which it is sending packets to that host. It is a special Internet Control Message Protocol (ICMP).

Ideally, a receiving host would detect when packets were stacking up too fast and send a source quench in time to slow the pace down so that no packets were lost. Note that the Internet Protocol of which ICMP is a part does not itself guarantee the delivery of packets. Higher-level protocols, such as TCP, have responsibility for ensuring successful end-to-end communication. IP and ICMP simply report errors or situations as they are detected so that packets can be resent or sent at a different pace. Source quench is not the only way to control flow in a network and not necessarily the most efficient way. In IP Version 4 (the most commonly-used IP version), routers are not allowed to originate a source quench and are not obligated to act on a received source quench. Because the source quench message may itself increase network traffic, other approaches to network flow control are preferred.

2.5 Survey of Wireless TCP Enhancements

Enhancements proposed in the literature usually fall into two categories. The first category attempts to hide wireless losses [2] from TCP so existing TCP algorithms do not need to change. This category can be further divided into Split Connection Approach, Link Layer Protocols, and Transport Layer Protocols.

The second category conveys wireless losses explicitly to TCP and lets TCP decide what congestion action to take. In this case, modifications to TCP are needed.

2.5.1 Split Connection Approach

The split connection approach splits the entire path into a wired connection and a wireless connection. TCP/IP runs independently on the two connections.

[2] We call packet losses caused by wireless transmission error "wireless losses", and packet losses caused by network congestion "congestion losses".

I-TCP [5] is one of the early protocols to split the entire path into two separate connections, one between the sender and the base station, and the other between the base station and the mobile host. Both connections run TCP/IP independently. The base station simply copies packets between the two connections. Errors occurring in the wireless connection do not affect the wired connection, so the performance will not be affected by wireless errors.

This solution is simple. Each connection is homogeneous and no modification is needed at the fixed host and the mobile host. Its problem is that acknowledgments received by the sender do not mean the packets have been received by their destination. When the mobile host moves to another cell, some acknowledged packets may get lost. Thus this method violates TCP's end-to-end semantic. The second drawback of I-TCP is heavy buffering at the base station. Large number of packets may pile up at the base station before they can be transported to the mobile host.

The basic model for **Two Connections** [7] is that the network is wired and the last hop from the base station to the mobile host is wireless. It assumes that the mobile host receives all its packets from the base station. TCP at the sender is assumed to know that the destination host is a mobile host.

When a fixed host wants to communicate with a mobile host, it opens two connections, one with the mobile host and the other with the base station. The second connection is a control connection used to estimate the congestion status on the wired network. Packets of these two connections are expected to be routed in the same way and hence are affected the same by the congestion. If the two connections experience the same fraction of packet loss, then the sender regards the packet loss as due to congestion and invokes congestion

control. Otherwise, it regards the packet loss as due to wireless errors, the lost packet is retransmitted, but congestion control is not invoked.

Under this scenario, the sender sends packets on the control connection in regular intervals sufficiently spaced so as not to cause overhead. The sender then periodically compares the fraction of acknowledged packets on the two connections and checks if the packets are lost due to congestion or due to transmission error on the wireless link. If the acknowledged fraction is significantly different in these two cases, then it concludes that the error in the wireless link is causing the packets to be dropped so the sender does not apply congestion control, does not reduce the window size and continues the increment as before. If the two fractions are the same, then the congestion control methods are applied as in normal TCP. When the packets are lost, TCP at the sender has to wait for a timeout after which it retransmits the packets. When the mobile host learns about this loss, it sends duplicate ACKs. On seeing the duplicate ACKs the sender knows that its previous packets have been lost and immediately resends the packets without waiting for the timeout.

This method has several severe defects. First, the assumption that the two connections will have the same route is questionable. Also, the correlation of packet losses between the two connections may be very low. Finally, this method adds a significant overhead of traffic to the network.

2.5.2 Link Layer Protocols

The second approach is **link layer protocols**, which enhance the wireless link by forward error correction and local retransmission to provide a reliable link to the upper layer, at the expense of introducing variable delays in data delivery. These methods naturally fit into the layered structure of networking protocols.

Xylomenos [75] argues that link layer solutions are applicable to any link and network topology. Its main problem is the interference of retransmission mechanisms at different layers. The variable delay caused by link layer retransmission may trigger a timeout at the source and start an end-to-end TCP retransmission. In addition, deciding the needed level of enhancement for a particular wireless link may be difficult. A single solution cannot address the needs of multiple protocols on different links.

2.5.3 Transport Layer Caching Protocols

The third approach is transport layer caching protocols, with the Snoop protocol [34] as the best-known implementation. This approach assumes that an intermediate node or an element in the wireless link knows that the transport layer is TCP and the underlying link is a wireless link. It selectively intercepts and modifies the data and acknowledgment traffic to mitigate the effect of wireless losses.

Snoop protocol [34] introduces a snooping agent at the base station to cache and compare TCP packets going out to the mobile host, as well as acknowledgments coming back. The snooping agent is able to determine which packet is lost on the wireless link and will schedule a local retransmission for the lost packet. At the same time, duplicate acknowledgments corresponding to the wireless loss are suppressed to avoid triggering a fast retransmit from the source.

Unlike other proposals, the Snoop protocol can find out the exact cause of packet losses and take actions to prevent the TCP sender from making unnecessary slowdowns. Snoop protocol requires the base station to cache TCP segments and keep per-connection state, which imposes a heavy buffering and processing burden on the base station. In addition, the Snoop agent at the base station needs to check the TCP header to find the sequence

number. If the packets are protected with IP security, the entire TCP packet is encrypted so intermediate routers including the base station are prohibited from reading the TCP header. Therefore, Snoop protocol will not work with encrypted traffic. Furthermore, Snoop protocol only works for traffic from the fixed host toward the mobile host. When the mobile host is the sender a different protocol has to be used.

Delayed Duplicate Acknowledgments (DDA) [73] is another transport layer protocol. Unlike the Snoop protocol, DDA does not need the involvement of the base station. It only needs minor modification at the receiver, and works with encrypted traffic. DDA assumes that a link level retransmission scheme is implemented. When out-of-order packets are received, the receiver sends duplicate ACKs for the first two out-of-order packets. If it gets more of them, the receiver defers further duplicate ACKs for a period. If during this period, the next in-sequence packet arrives, the receiver discards the deferred duplicate ACKs and sends a new ACK. If the in-sequence packet does not arrive during this period, the receiver releases the deferred duplicate ACKs to trigger a retransmission. DDA is a simple proposal and requires modification only at the mobile host. However, it does not distinguish between wireless and congestion losses, and increases the delay of retransmitted congestion losses.

The basic idea of **Fast-TCP** is that a network element, such as a router, delays the IP packets carrying TCP acknowledgments when congestion is about to occur. Since the TCP source does not receive an acknowledgment, it keeps its current transmission window until the delayed acknowledgment is received.

It has been proved that Fast-TCP can speed up TCP flow control time, reduce buffer oscillation, increase bandwidth utilization, increase throughput and reduce packet losses in IP networks with wired links.

Fast-TCP can prevent buffer overflow during slow start when the TCP transmission window increases rapidly. Since wireless links cause lost segments and as a consequence, slow starts, the Fast-TCP has even more opportunity to act. Naturally, the overall amount of data transferred is higher with the wired link.

The main conclusion is that, even without any special adaptations to wireless protocols, Fast-TCP improves the performance of wireless TCP connections.

Multiple Acknowledgments [8] uses two types of acknowledgments to distinguish between packet losses in the wired network and in the wireless link. The partial acknowledgment ACKp is sent by the base station to tell the sender that it has received the packet. The complete acknowledgment ACKc is sent by the mobile host.

Local retransmission is implemented on the wireless link. When the base station receives a packet, it sends an ACKp to the sender and delivers the packet to the mobile host. If the local retransmission timer expires, it retransmits the packet. ACKs from the mobile host are forwarded only if they are needed by the sender TCP. ACKs that trigger unnecessary retransmissions are discarded. When the sender receives an ACKp, it marks the packet as having been received by the base station. Only when the packet is not received by the base station will the packet be retransmitted and the congestion control actions be taken.

This method can determine the cause of packet losses, at the cost of extra traffic to send the partial acknowledgments. However, its main problem is the modification at the fixed host, which is usually not under the wireless service provider's control. Even if the wireless service provider can make changes in some fixed hosts, the mobile host is limited to visiting these hosts only.

2.5.4 Explicit Notification Protocols

Unlike other methods that hide wireless errors, explicit notification protocols explicitly communicate the cause of packet loss to the source so TCP can take correct congestion control actions.

Explicit Loss Notification (ELN) [6] is a general mechanism by which the cause of the packet loss can be communicated to the TCP sender. A snooping agent running at the base station monitors all TCP segments that arrive over the wireless link. It does not cache TCP segments since it does not perform any retransmissions. Rather, it keeps track of holes in the sequence space. These holes correspond to segments that have been lost over wireless links.

When an ACK arrives from the receiver, the agent at the base station consults its list of holes. If the ACK corresponds to a segment in the list, the ELN bit in the ACK is set before being forwarded to the data sender. When the sender receives an ACK with ELN bit set, it retransmits the next segment but does not take any congestion control actions.

In ELN, the base station suffers heavy processing burden just as in the Snoop protocol. It works only when the mobile host is the sender. In addition, ELN introduces unnecessarily long delays for the retransmitted packets. When the base station detects a wireless error, it does not request a retransmission immediately. Instead, it waits for the corresponding duplicate ACKs coming back and sets the ELN bit to trigger the retransmission. This adds almost one entire round trip delay to the retransmitted packets.

The **Inter-arrival Time** method proposed in [9] again uses the model where the network is wired and the last hop is wireless. It also assumes the wireless link is the bottleneck in the network. Since the wireless link is the bottleneck, most packets are queued at the

base station and transmitted back to back on the wireless link. The packet inter-arrival time is defined as the time between arrival of consecutive packets.

When no packet is lost, the packet inter-arrival time is the same as the time required to transfer the packet on the wireless network, denoted as T. If packet n is lost due to wireless error, the inter-arrival time between packets $n-1$ and $n+1$ is $2T$. If packet n is lost due to congestion, the packets $n-1$ and $n+1$ will become back-to-back in the base station's queue. Therefore, their inter-arrival time at the mobile host is T.

Let T_{min} be the minimum inter-arrival time observed so far. P_o denotes the out-of-order packet received by the receiver. P_i was the last in-sequence packet received before P_o. T_g is the time between arrivals of P_o and P_i and let n be the number of packets missing between P_i and P_o. If $(n+1)T_{min} \leq T_g \leq (n+2)T_{min}$ then n missing packets are assumed to be lost due to wireless transmission errors otherwise they are assumed to be lost due to congestion.

This method is based on unrealistic assumptions on the network. If there is another mobile user at the same cell, the back-to-back assumption will be broken. Therefore, this method is not feasible.

If the TCP sender has accumulated a certain number of dupacks and is confident that the next unacknowledged packet is lost, it can invoke the Fast Retransmit algorithm immediately. The number of dupacks that should be accumulated is a function of the number of unacknowledged packets through the **Segment-In-Flight Estimate** [60].

A new variable $sifest$, which is the sender's estimation of the amount of in-flight segments and is zero initially, is proposed to be added at the sender for every active TCP session. When a new data packet is sent, $sifest$ is increased by one. When the TCP sender gets a $newack$, $sifest$ is decreased by the number of acknowledged segments. If timeout

occurs or the sender has been idle for more than one rtt, $sifest$ is zero again since the sender is going to reprobe the network. If the TCP sender retransmits a sent packet triggered by schemes other than a timeout, $sifest$ remains untouched, since it is assumed that the previous copy of this packet has already left the network. The receiver and intermediate systems are unmodified, and the modified TCP can be deployed incrementally and interact with ordinary TCP compatibly over the Internet.

When the TCP sender already has $(sifest - 1)$ dupacks, or dupacks have exceeded a fixed threshold, and the most unacknowledged packet has not been resent in the last rtt, this packet is resent immediately according to the Fast Retransmit. If more new data packets are allowed by *cwnd* and the *cwnd inflating* algorithm, they can be piped into the networks to trigger more ACKs and to help TCP endpoints regain the *self-clocking* earlier. Integrating the SIF estimation with ordinary TCP variants is quite straightforward and conflict-free with the original algorithms. With the SIF enhancement, the TCP sender might become less tolerant on re-ordered packets, and the estimation error might be accumulated for a while. However, when SIF is effective, since there are only few in-flight segments and the estimation is re-initialized periodically, the probability of *dupack* occurrence due to packet reordering is negligible.

2.6 Survey of Media Access Control

The most important interface in wireless networks is the radio access interface between the mobile stations and the base station. The wireless radio link enables mobile stations to be connected to the network while moving. The air media is shared by the base station and all mobile stations in the cell. The access to the media is coordinated by the MAC protocol.

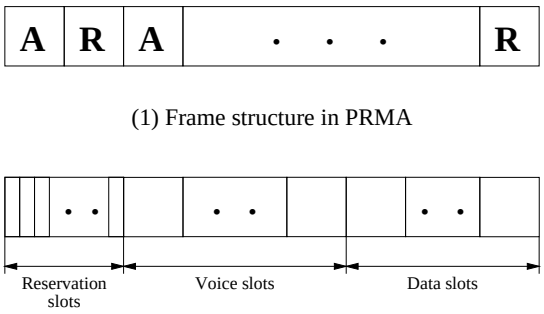
As more real-time applications are deployed on mobile stations, the support for real-time application becomes an important requirement for MAC protocols. In this section, we summarize existing MAC protocols that support real-time applications on wireless local area networks.

2.6.1 Packet Reservation Multiple Access

The first MAC protocol proposed to support wireless communication is the Packet Reservation Multiple Access (PRMA) protocol [59]. As shown in Fig. 2.3 (1), in PRMA time is divided into basic transmission units called *slots*, and several slots form a scheduling unit called a *frame*. Users are classified into two types: voice users and data users. PRMA does not have dedicated reservation bandwidth. Each user with voice/data to transmit randomly chooses a slot available inside a frame for packet transmission. Whether or not contention occurs in a slot will be known to all the senders by the end of the slot. If a voice user successfully transmits its voice packet in a slot, the slots will be labeled as reserved in subsequent frames until released by the voice user upon completion of its transmission. This rule, however, does not apply to data users, who are required to contend for every slot it would like to use for data transmission. Due to its CSMA nature, PRMA suffers from low utilization in medium to heavy traffic loads.

2.6.2 Dynamic Time Division Multiple Access

Dynamic Time Division Multiple Access (D-TDMA) [74] was proposed by Wilson etc in 1993. As shown in Fig. 2.3 (2), in D-TDMA, time is divided into frames, and each frame is composed of reservation slots, voice slots and data slots. A slotted ALOHA approach is used in reservation slots for reservation requests. That is, to reserve an information (voice/data) slot, a user sends a reservation packet in a randomly chosen reservation slot.



(1) Frame structure in PRMA

(2) Frame structure in D-TDMA

Figure 2.3: Frame structures for the PRMA and D-TDMA protocols.

The reservation packet contains information needed to establish a connection (e.g., the source/destination addresses). At the end of a reservation period, successful reservation will be identified and the final slot schedule will be broadcast to all the users by the base station. Once allocated a voice slot, a user can use the same slots in subsequent frames until it completes its transmission, while a data user is granted one data slot (in the same frame as it makes the reservation) at a time. Unsuccessful users will retry in the next frame according to a *reservation retransmission* probability.

2.6.3 Resource Auction Multiple Access

Resource Auction Multiple Access (RAMA) [3] is very similar to D-TDMA except that it uses a different reservation approach. As shown in Fig. 2.4, reservation slots in a frame are replaced by auction slots in RAMA. In each auction slot, the available resources (i.e., information slots) will be auctioned to requesting users and will be assigned to the winner of the auction. The auction procedure works as follows (Fig. 2.4): each requesting user is assigned a user ID which is randomly generated when the user decides to attend the auction. The number of digits used in the random number depends on the number of users currently

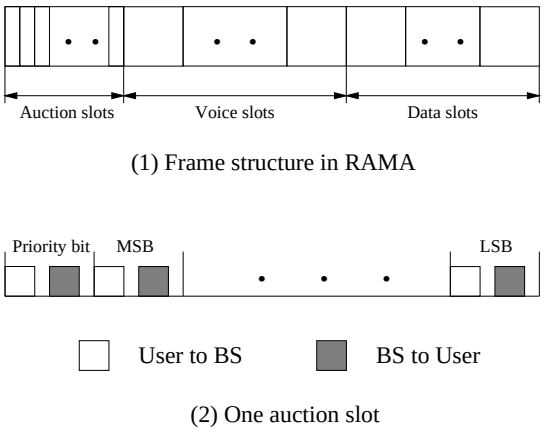
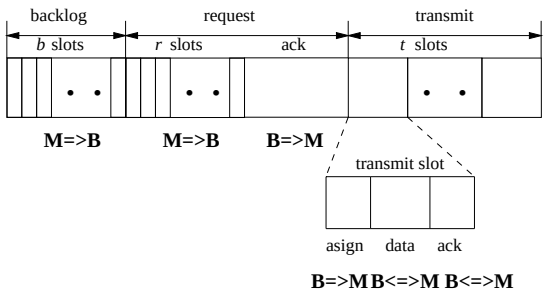


Figure 2.4: Frame structures of RAMA

in the network. Requesting users start to transmit their IDs in the auction slots, one at a time, from the most significant bit to the least significant bit. After each bit transmission, the base station broadcasts the largest bit value it just received, and those contending mobile hosts with unmatched bit value will drop off. There will be a final winner by the end of each auction slot. This winner will not attend any future auction in the same frame. The users dropping off in the current auction slot can select another random number and re-enter the auction in the next slot. One attractive property of RAMA is that it is guaranteed that one mobile host will finally win out in each auction and be successful in sending its reservation request to the base station.

2.6.4 Dynamic Reservation Multiple Access and Floor Acquisition Multiple Access

DRMA [64] is a variation of the above protocols, and differs in the degree of design complexity and the level of bandwidth efficiency thus achieved. DRMA eliminates the reservation/auction slots in D-TDMA/RAMA, and uses (if necessary) an available slot as a



Frame structure in RQMA

Figure 2.5: Frame structures of RQMA.

set of reservation slots. Efficiency is achieved by dynamically assigning reservation slots, rather than using fixed reservation slots. FAMA [19], on the other hand, basically applies the carrier sense multiple access with collision detection mechanism to the control and jamming packets sent from mobile hosts to the base station, and can be regarded as a CSMA/CD scheme in a wireless LAN.

All the above wireless MAC protocols are tailored to meet the specific requirement of supporting only voice and data users, and do not address the need for supporting other aspects of QoS. In particular, they provide no temporal QoS required by bus location tracking applications. Recently, several other MAC protocols have been proposed that address the QoS issues in wireless LANs, which we summarize below.

2.6.5 Remote-Queuing Multiple Access

RQMA [25] supports three types of traffic: constant bit rate (CBR), real-time, and best-effort. A frame in RQMA is divided into three fields: b backlog slots, r request slots (and their corresponding ack subfields), and t transmission slots (Fig. 2.5). A mobile host

sends a request in a request slot (in a slotted ALOHA fashion) to the base station to either establish a RT/CBR session or send best-effort packets. If the base station successfully receives a request in a request slot, it sends an ack in the corresponding ack subfield. Once a real-time session is established, a mobile host then uses one backlog slot (assigned by the base station) to inform the base station of any newly-arrived packets of the real-time session and their deadlines. The base station then determines when a mobile host can send/receive data packets by specifying in the *assign* subfield of each transmit slot the mobile host id and the session id. The most desirable feature of RQMA [25] is that it takes into consideration of the error characteristic of the wireless channel, and establishes *a priori* a *real-time retransmission* session to retransmit time-critical data packets upon error detection in the normal transmission phase.

2.6.6 Data Over Cable Service Interface Specification

It has also come to our attention that the Cable Modem standard — Data Over Cable Service Interface Specification (DOCSIS) [15] — has recently been developed by the industrial association Multimedia Cable Network System (MCNS) Partners to specify the internal and external network interfaces for a system that allows bi-directional transfer of Internet Protocol traffic between the cable system and the customer ends over a cable television system. In particular, the Radio Frequency (RF) Interface Specification of DOCSIS [15] specifies the MCNS MAC protocol to be used in the Cable Modem system.

CHAPTER 3

PROBLEM STATEMENT AND METHODOLOGY

Enhancements of wireless networks involve many aspects of the network. The focus of our study is on the transport layer and the media access control layer, including algorithms at the end points and support from network routers. The issues resolved in this study include:

- Identifying requirements on wireless TCP enhancement
- Analyzing wireless TCP performance degradation
- Designing a wireless TCP enhancement scheme
- Delivering faster congestion feedback
- Allocating resources in asymmetric wireless LANs
- Balancing delay and bandwidth efficiency in voice trunking

These issues and the methodology to address these issues are discussed below.

3.1 Identifying Requirements on Wireless TCP Enhancement

While many proposals have been suggested for wireless TCP enhancement, many of them pose unrealistic assumptions and need modifications in places that are essentially impossible. Some proposals apply only to a special network configuration, and some only solve the performance degradation problem for traffic in one direction.

In this part of the study, we identify and prioritize requirements that are desirable in practice. The goal of formulating this requirement list is to decide criteria to evaluate enhancements and to find new enhancement approaches.

3.2 Analyzing Wireless TCP Performance Degradation

It has been widely recognized that TCP has degraded performance in wireless networks. A common misunderstanding in the research community is that the performance degradation is caused by TCP's inability to distinguish between wireless and congestion losses. Enhancements proposed in the literature focus on distinguishing wireless packet losses from congestion losses.

However, through simulations we found these enhancements do not work well, even though the cause of packet loss has been communicated to the TCP sender and congestion control actions at the sender have been suppressed or duplicate acknowledgments have been discarded. The simulation results strongly suggest that some causes other than the inability to distinguish between the two types of losses are degrading the performance.

In order to find the contributors of the performance degradation, we designed an experiment to collect TCP performance data under different loss scenarios, and use stepwise multiple regression to analyze the leading contributors of TCP performance degradation.

The goal of this analysis is to find the leading contributors of performance degradation in wireless TCP, and to show that current proposals did not solve the degradation problem. The conclusion of this analysis also suggests the direction of future enhancement efforts.

3.3 Designing Wireless TCP Enhancement Scheme

As indicated by the result of the previous two issues, current wireless TCP enhancements are not satisfactory. A new enhancement scheme is needed. The goal of this scheme is to achieve compatibility with current TCP implementations, make modifications to current network implementations only in the scope of wireless service providers, and most importantly, to solve the performance degradation problem caused by timeout.

The proposed scheme is based on the ECN protocol that was standardized by IETF in RFC 2481 and is expected to be widely implemented. This scheme makes use of the sequential coherence of ECN packet marking which is based on the fact that congestion does not happen nor disappear suddenly. Before the congestion reaches to the level of buffer overflow, some packets must be marked. Similarly, after a packet is dropped, some subsequent packets will be marked. Congestion coherence can help to distinguish the two types of packet losses, and ECN can help to eliminate timeouts arising from multiple losses in one window.

3.4 Delivering Faster Congestion Feedback

Delivering congestion feedback in time is critical to the success of congestion control algorithms. The faster the congestion feedback is received by the sender, the more effective the congestion action will be. Historically, timeout [41], triple duplicate acknowledgments [70] and explicit congestion notification [26] have been used to deliver congestion feedback

to the sender. Each of these three mechanisms is faster than its predecessor and brings better TCP performance. In this study, we find that marking the packet in the front of queue can deliver even faster congestion feedback to the source.

In this research, the performance effectiveness of the mark-front strategy is studied through analysis and simulation. The improvement of mark-front strategy includes smaller buffer size requirement at the routers and higher TCP throughput. It also avoids the lock-out phenomenon, and improves the fairness among connections.

3.5 Allocating Resources in Asymmetric Wireless LANs

Media Access Control protocol is one of the most important protocols in a wireless LAN. A vast variety of MAC protocols were proposed in the literature in order to address the various needs of different wireless networks. The OSU-MAC is proposed to support both real-time and non-real-time applications in the OSU narrow-band wireless modem testbed, subject to its physical layer characteristics and constraints.

The OSU wireless modem testbed is a narrow-band modem with half-duplex transmission and asymmetric channels. The base station, which is equipped with a more sensitive transceiver and having higher power consumption, can transmit and listen at the same time. The mobile station, with its lighter weight handset and small battery, can only transmit or listen at any given time. Moreover, switching from transmitting mode to listening mode and vice versa requires a guard time. Because of the different equipment and power consumption, the forward and reverse channels have different data and error rates. In addition, in a multi-user environment, users can join and leave the cell at any time and compete for the media access.

The half-duplex media, asymmetric channels and real-time applications create great challenges in the media access design. In the design of OSU-MAC, a number of unique techniques have been used, including double control fields, dynamic registration and reservation, separation of real-time slots and non-real-time slots, two levels of error correction code, etc.

The goal of OSU-MAC is to provide support for both real-time and non-real-time applications, and to maximally improve the media efficiency under the physical layer constraints. The simulation results of the MAC protocol prove that the MAC scheme does improve the efficiency of the media.

3.6 Balancing Delay and Bandwidth Efficiency in Voice Trunking

Delay and bandwidth are two important aspects of Quality of Service for real-time applications. Real-time applications, especially voice, are sensitive to delay. Human ears are very good at picking up the delay in voice. In voice conversation, most people notice round trip time when it exceeds 250 ms. ITU-T G.114 recommends 150 ms as the maximum desired one-way latency to achieve high-quality voice. Beyond this latency, callers begin to feel uneasy holding a two-way conversation and usually end up talking over each other. At a 500 ms or larger round trip time, phone calls are impractical.

The problem of delay becomes more severe when more efficient compression/decompression methods are used. For example, in order to fill an ATM cell which has 48-byte payload, the ITU-T G.711 (64 kbps) codec needs 6 ms, but the more efficient ITU-T G.723.1 (5.3 kbps) codec needs 72 ms. Notice that this 72 ms does not include the propagation delay, queueing delay, etc, that the cell must undergo when it travels through the networks.

ATM Adaptation Layer 2 (AAL2) has been designed to reduce the packing delay. It is described in ITU-T Recommendation I.366.2[38] and ATM Forum specification “ATM Trunking using AAL2 for Narrow-band services ”[4]. The idea is to multiplex voice packets from multiple sources into one ATM cell so that the time to fill a cell can be reduced.

However, there is a balance between the delay and the bandwidth. In AAL2, _CU that decides the expiration of the packing. A larger is a parameter Timer Timer _CU value means an ATM cell can wait longer and carry more voice packets, a smaller Timer_CU value implies that more cells might be sent before fully packed in order to reduce the delay incurred by elongated packing process. Therefore the Timer _CU value is an important parameter to balance voice delay and bandwidth efficiency.

The goal of this research is to establish a stochastic model to describe packing ve principles and an algorithm to set the timer value.

r Summary

n, we have identified problems of current wireless networks and defined e can improve. The areas include (1) Identifying requirements on wireless nt, (2) Analyzing wireless TCP performance degradation, (3) Designing enhancement scheme, (4) Delivering faster congestion feedback, (5) Alles in asymmetric wireless LANs, and (6) Balancing delay and bandwidth ce trunking, The next six chapters of this dissertation are devoted to these

CHAPTER 4

REQUIREMENTS OF WIRELESS TCP ENHANCEMENTS

TCP is known to have poor performance over wireless links. With the rapid development of wireless and mobile communications, there is a pressing need to enhance TCP performance over wireless links. In the past few years a number of enhancements have been proposed, but many of these enhancements are not practical in terms of assumptions, implementation and applicability. As criteria to evaluate these enhancements, in this chapter we establish a set of requirements for wireless TCP enhancements.

4.1 Implementation Requirements

Although the proposals reviewed in Chapter 2 solve the poor performance problem in some ways, they are based on questionable assumptions, need modifications at some points, and apply only to certain traffic configurations. Here we summarize a set of requirements that we think are desirable in practice, listed in the order of importance.

4.1.1 Semantic Requirement

First, **the enhancement must maintain the end-to-end semantic**. TCP is a *reliable* transport protocol. All packets acknowledged must have been received correctly at their

destination. Any enhancement not meeting this requirement is *not* a reliable transport protocol.

4.1.2 Local Modification Requirement

Second, **all modifications to the existing algorithm must be local**. Adding a wireless link to the network should not require the entire network to change. In the case of a mobile host, only the base station and the mobile host are under the control of the wireless service provider. Modifications beyond this scope are usually impractical. For example, when a wireless service provider offers an Internet service, he can modify the base station or the mobile station to improve TCP performance, but requiring all web sites that his subscribers may visit to change is impractical.

4.1.3 Encryption Requirement

Third, **the enhancement must apply to encrypted traffic**. As data security gets more important, a large portion of traffic is likely to be encrypted. Encryption protocols like IPSEC encrypt IP payload so that intermediate nodes cannot see what is being carried. Only the destination has access to the content of IP packets. Enhancement solutions that monitor the TCP header at the base station do not meet this requirement.

4.1.4 Two-way Traffic Requirement

Fourth, **the enhancement should apply to two-way traffic**. Many enhancements are designed for mobile receivers, and work only for traffic from the wired network to the mobile host. However, in almost all cases the mobile hosts also send data packets. Traffic in both directions is affected by wireless errors. In order to fully utilize the wireless bandwidth, the enhancement must work for both directions.

4.1.5 Intermediate Link Requirement

Fifth, **the enhancement should apply to intermediate wireless links**. Many enhancements presume the wireless link is the last hop of the connection and work only for this configuration. While this configuration is common, other types of networks like ad-hoc networks, mobile networks and satellite links have intermediate wireless links. It is desirable that the enhancement can apply to intermediate wireless links.

In addition to the above five requirements, it is desirable to control the traffic, buffering and processing overhead.

	I-TCP	**Multiple Acks**	**Control Connection**	**Snoop**	**ELN**	**Delayed Dupacks**
Modify MH	no	no	yes	no	no	yes
Modify BS	yes	yes	no	yes	yes	no
Modify FH	no	yes	yes	no	no	no
Complexity	simple	moderate	moderate	high	high	simple
Add traffic	no	yes	yes	no	no	no
BS buffering	heavy	heavy	no	heavy	no	light
TCP states at BS	light	light	no	heavy	heavy	no
Two-way solution	yes	no	no	no	no	no
End-to-end semantic	no	yes	yes	yes	yes	yes
Interpret losses	no	yes	probably	yes	yes	no
Interpret dupacks	no	yes	probably	yes	yes	yes
Delay variations	small	small	large	small	large	small

Notes:

1. BS buffering: high if transport layer retransmission; light if link layer retransmission; no if retransmission is not needed.
2. TCP states at BS: high if BS needs to maintain a hole list; light if only window variables are needed.
3. Delay and delay variation: large if the corrupted packets are retransmitted end-to-end; small if retransmitted locally.

Table 4.1: Comparison of proposals to enhance TCP over wireless links

4.2 How Current Enhancements Meet the Requirements

Current enhancement proposals in the literature have been reviewed in Section 2.5. Table 4.1.5 summarizes the major characteristics of these enhancements, including where the modifications are made and their complexity and performance.

Table 4.2 summarizes whether the current proposals meet the requirements and their traffic, buffering and computing overhead. A $\surd$ means the requirement is met or the overhead is low, a $\times$ means the requirement is not met or the overhead is high. Surprisingly, none of current solutions meet all the requirements.

	I-TCP	**Multiple Acks**	**Control Connection**	**Snoop**	**ELN**	**Delayed Dupacks**
Semantic Requirement	$\times$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$
Local Requirement	$\surd$	$\times$	$\times$	$\surd$	$\surd$	$\surd$
Encryption Requirement	$\times$	$\times$	$\surd$	$\times$	$\times$	$\surd$
Two-way Requirement	$\surd$	$\times$	$\times$	$\times$	$\times$	$\times$
Intermediate Link Rqmt	$\surd$	$\times$	$\surd$	$\times$	$\times$	$\surd$
Traffic Overhead	$\surd$	$\times$	$\times$	$\surd$	$\surd$	$\surd$
Buffering Overhead	$\times$	$\surd$	$\surd$	$\times$	$\times$	$\surd$
Computing Overhead	$\surd$	$\surd$	$\surd$	$\times$	$\times$	$\surd$

Notes: Both Snoop and ELN are one-way solutions, but they can be combined to provide a two-way solution.

Table 4.2: Which solutions meet the requirements?

Based on the classification in Section 2.5 and the requirements in this chapter, we find that the split connection approach violates the end-to-end semantic. Explicit notification protocols are not local. The link layer protocols are highly dependent on the characteristic

of the wireless link and a general method is difficult to find to for all links. Therefore, our study mainly concentrates on the transport layer protocols.

CHAPTER 5

LEADING CONTRIBUTORS OF WIRELESS TCP PERFORMANCE DEGRADATION

Transmission errors on wireless links cause end-to-end retransmissions and unnecessary slowdowns in wireless TCP. A common understanding in the research community is that the performance degradation of wireless TCP is caused by TCP's inability to distinguish between wireless and congestion losses. In the past few years, a number of enhancements have been proposed to improve TCP performance over wireless links. These enhancements implement local retransmission to avoid end-to-end retransmission, and develop schemes to distinguish the two different types of packet losses. In this chapter, we conduct a number of simulations to collect performance data, and statistically analyze the contributors of wireless TCP performance degradation.

5.1 Introduction

TCP was designed mainly for wired networks, where transmission errors are rare and the majority of packet losses are caused by congestion. An underlying assumption of TCP congestion control algorithm is that packet losses and the resulting timeout at the source are indications of network congestion and the source should reduce their traffic on timeout [41].

When TCP is deployed on wireless networks, packet losses due to transmission error are retransmitted from the source, and upon retransmission the effective transmission rate is reduced by half. If wireless errors happen frequently, the effective transmission rate of the wireless link could become very small. Therefore, TCP has severe performance degradation on wireless links.

A common understanding in the research community is that the performance degradation is caused by the lack of local retransmission on the wireless link and by TCP's inability to distinguish wireless losses and congestion losses. Enhancements proposed in the past few years focus on implementing local link layer retransmission and developing schemes to distinguish the two types of packet losses. However, our simulations show these enhancements do not work well in many cases. We find that retransmission and unnecessary slowdown are not the only contributors of performance degradation. Wireless TCP suffers heavily from timeout. Timeout, which is the result of multiple losses in one window, happens only occasionally in wired networks. With the presence of wireless links, the chance of having multiple packet losses in one window is significantly increased, and causes severe performance degradation.

In this chapter we design an experiment to collect TCP performance data under different loss scenarios, and use stepwise multiple regression to analyze the leading contributors of TCP performance degradation. Our result reveals that timeout is the leading and dominant contributor of performance degradation. In addition, our simulations show that current enhancements fail to improve TCP's timeout behavior and therefore exhibit lower performance in many cases. This study indicates that a good enhancement must change TCP's timeout behavior. Based on the conclusion of this study, a new enhancement approach is

proposed. Simulation results show that such an approach outperforms other enhancement proposals.

5.2 Contributors of TCP Performance Degradation

The TCP algorithm is designed to be conservative on the amount of outstanding data in the network. A congestion window (*cwnd*) is used to control the number of outstanding TCP segments in the network. At the beginning of transmission, *cwnd* is set to one segment. When the acknowledgment is received, *cwnd* is increased by one segment. If no packet is lost, TCP can transmit one segment in the first round-trip time (RTT), two segments in the second RTT, and four segments in the third RTT. This period is called the slow start phase. The transmission rate grows exponentially until *cwnd* reaches a slow start threshold (*ssthresh*) value or when a packet is detected to be lost. At the beginning, *ssthresh* is initialized to an arbitrary high value and is set to half of the current *cwnd* value if a packet loss is detected. After *cwnd* grows larger than *ssthresh*, the sender enters the congestion avoidance phase, during which *cwnd* increases by only one segment per RTT.

When too many packets are sent to the network, some packets will be lost due to congestion. In early versions of TCP, the receiver would only acknowledge the highest segment it had successfully received in order, so the source relied solely on retransmission timeout to recover from packet losses. The Reno version [2] of TCP allows the sender to retransmit a packet when three successive duplicate acknowledgments are received. This procedure is called fast retransmit. Fast retransmit can recover only one packet loss in a window. When multiple losses happen in the same window, the source will timeout and a slow start procedure begins.

The above TCP congestion control mechanisms were designed mainly for wired networks. With the presence of wireless losses in the network, the underlying assumptions of these congestion control mechanisms are completely changed. Through simulation and comparison, we find three contributors of TCP performance degradation: end-to-end retransmission, unnecessary slowdown and timeout.

First, when a packet is lost, it will be retransmitted either locally or end-to-end. In general, all congestion losses are retransmitted end-to-end. If a local retransmission is implemented on the wireless link, the majority of wireless losses can be retransmitted locally. Packets suffering a second loss in the retransmission may cause the TCP source to timeout and trigger an end-to-end retransmission. If the retransmission is end-to-end or if the bottleneck link is on the retransmission path, the extra traffic uses the bottleneck bandwidth and may reduce the goodput.

Second, when a wireless loss is retransmitted end-to-end, existing TCP algorithms treat the packet loss as an indication of congestion and slow down the transmission rate. Unnecessary slowdown causes significant performance degradation because the effective transmission rate is cut to half. This degradation can be avoided if a mechanism is found to distinguish between the two types of losses.

Third, existing TCP algorithms (Tahoe and Reno) can recover only one packet loss in a window using fast retransmit. Two or more packet losses (either congestion or wireless) in the same window usually result in timeout. Timeout causes severe performance degradation because after timeout it takes many round trip times (RTT) to bring the transmission rate to normal level. If packet loss is used for delivering congestion feedback, a wireless loss happening in the same window as a congestion loss is likely to cause timeout.

Different TCP versions and enhancements have slightly different treatment to packet losses, retransmission, slowdown and timeout, so the impact of these contributors on the performance may vary slightly.

5.3 Experiment Design and Data Collection

In order to find the performance impact of the three contributors, we conducted a set of simulations with the ns simulator [53], collected TCP performance data for different packet error rates and enhancements, and tried to analyze the data using multiple regression. Our purpose is not to find a formula for calculating degradation in general cases, but rather to find the leading contributors of TCP degradation and therefore give light to the direction of enhancement design.

The simulations were conducted on a simplified network model shown in Figure 5.1, where s_1, s_2 are the sources and d_1, d_2 are the destinations. The numbers beside each link represent its rate and delay. The link between intermediate routers r_1 and r_2 is the bottleneck link. The link between r_2 and d_1 is a wireless link.

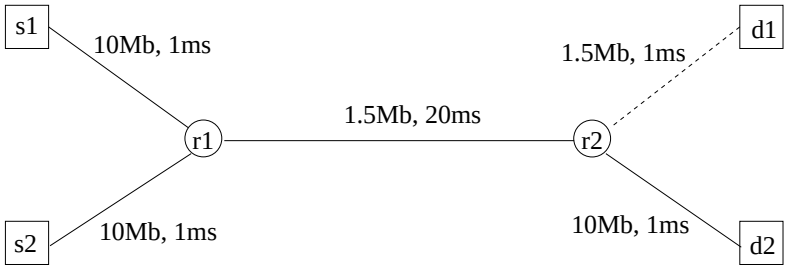


Figure 5.1: Simulation model

The experiment traffic is an FTP session from s_1 to d_1 using TCP Reno as the transport protocol. The background traffic is a UDP flow from s_2 to d_2 generated by an exponential on-off model. The mean burst period and the mean silence period are both 100 ms, and the burst data rate is 500 kbps. Both TCP and UDP packet sizes are 1000 bytes, and TCP acknowledgments are 40 bytes long.

Link layer retransmission is implemented on the wireless link. Packets sent but not acknowledged at the link level within 40 ms are resent. Retransmitted packets are also subject to wireless errors at the same rate. The waiting time for DDA is 81 ms so packets delivered within two retransmissions are accepted. The packet error rate of the wireless link is varied to test the performance of various proposals under different loss scenarios.

To reflect steady state measurement, the simulation time should be long enough to minimize the effect of the initial transient state. Longer simulation normally generates smoother aggregate results. Techniques to determine the simulation end time can be found in Chapter 25 of [42]. In our experiment, we tried various simulation times and found the results of 500 seconds show the essential features without noticeable difference from longer simulations, so all aggregate measurements are collected from 500-second simulations.

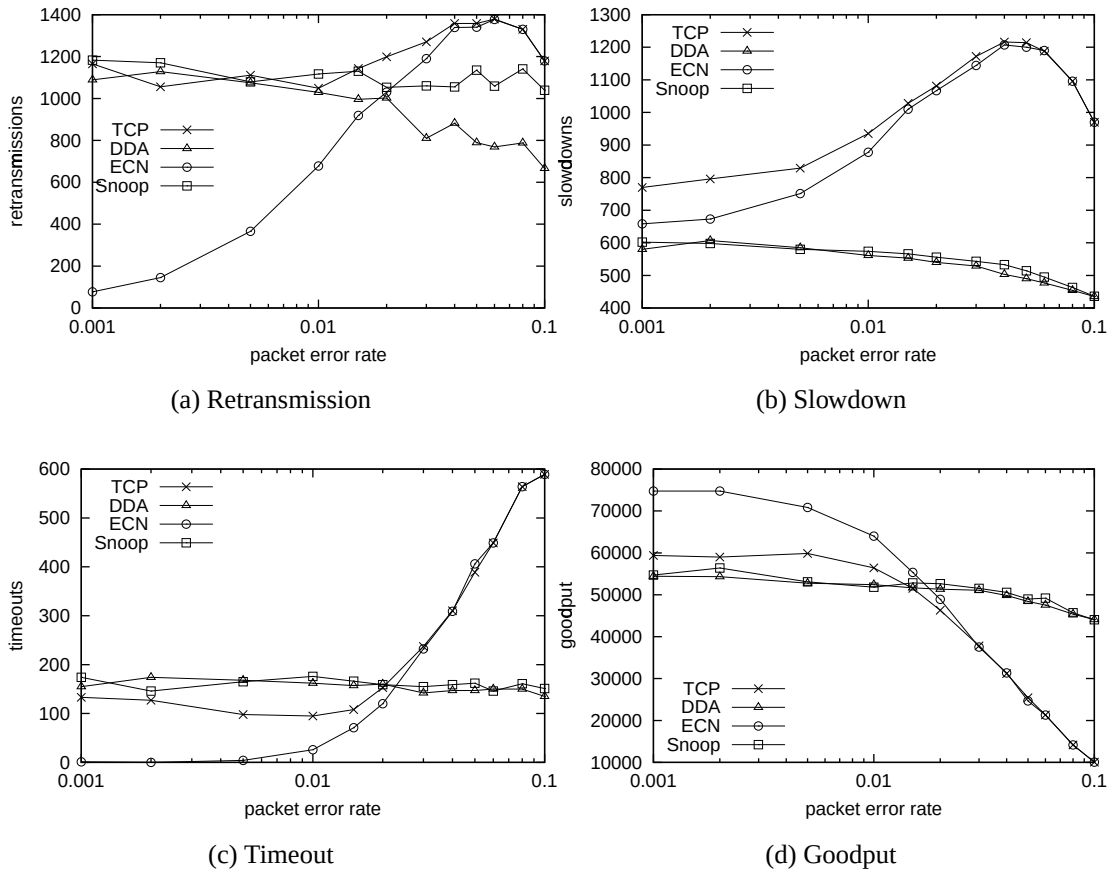
In each simulation, the following data are collected:

RETRANS: number of end-to-end retransmissions, including congestion losses, wireless losses that are retransmitted end-to-end, and packets retransmitted during timeout.

SLOWDOWN: number of slowdown actions taken at the source. The slowdown action can be triggered by duplicate acknowledgments or by ECN-Echo if ECN is used.

TIMEOUT: number of timeouts at the source.

GOODPUT: number of packets successfully received and acknowledged.



(a) Retransmission (b) Slowdown

(c) Timeout (d) Goodput

Figure 5.2: Retransmission, slowdown, timeout and goodput for different methods

The methods we simulated include plain TCP, DDA, ECN and Snoop. ECN (Explicit Congestion Notification) is not an enhancement for wireless network, but it improves TCP performance in certain scenarios

5.4 Regression Methodology and Results

The goal of our analysis is to find a set of variables $\{X_1, X_2, \ldots, X_m\}$ from RETRANS, SLOWDOWN and TIMEOUT such that they constitute a good regression for the goodput.

$$goodput = b_0 + b_1 X_1 + b_2 X_2 + \cdots + b_m X_m \tag{5.1}$$

By studying the regression model, we hope to find the leading contributors of TCP performance degradation.

The difficulty in this regression comes from the intercorrelation among variables. The three independent variables — RETRANS, SLOWDOWN and TIMEOUT — affect each other. Adding or removing a variable from the model significantly affects the coefficients. Stepwise variable selection is a frequently used procedure to select the minimal set of independent variables to constitute a satisfactory regression, especially when the candidate independent variables have strong correlation.

In each step of the stepwise selection, a variable is added to or removed from the regression model (5.1). The first variable entered at step 1 is the one with the strongest positive (or negative) simple correlation with the dependent variable. At step 2 (and at each subsequent step), the variable with the strongest partial correlation is entered. At each step, the hypothesis that the coefficient of the entered variable is 0 is tested using its F statistic. Stepping stops when an established criterion for the F no longer holds.

The entire selection procedure is carried out using the SPSS software. The results are presented in Table 5.1 to 5.6.

Table 5.1 characterizes the mean and standard deviation of the variables, which are defined respectively as

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \quad \text{and} \quad s_x = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}} \tag{5.2}$$

Here $n = 204$ and x_i is the observed value in the i-th data entry.

	Mean	**Std. Deviation**	**N**
GOODPUT	51340.19	13820.967	204
RETRANS	996.39	316.490	204
SLOWDOWN	744.35	234.828	204
TIMEOUT	167.57	127.428	204

Table 5.1: Descriptive Statistics

The correlation between two variables x and y is defined as

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{(n-1)s_x s_y} \tag{5.3}$$

The correlation among the collected variables is listed in Table 5.2. This table reveals that TIMEOUT has the strongest correlation with GOODPUT, and the correlation level among these variables is pretty high.

	GOODPUT	RETRANS	SLOWDOWN	TIMEOUT
GOODPUT	1.000	-.710	-.504	-.952
RETRANS	-.710	1.000	.317	.680
SLOWDOWN	-.504	.317	1.000	.469
TIMEOUT	-.952	.680	.469	1.000

Table 5.2: Correlation

In the stepwise selection, the criterion to enter a variable is that the probability of F statistic is smaller or equal to 0.05; the criterion to remove a variable from the model is that

the probability of F statistic is greater than or equal to 0.10. Table 5.3 records the order that the variables are entered or removed from the model.

Model	Variables Entered	Variables Removed
1	TIMEOUT	—
2	RETRANS	—
3	SLOWDOWN	—

Table 5.3: Variable Entered/Removed

Table 5.4 summarizes how well each regression model fits the observed data. R, the coefficient of multiple regression, is the correlation between the observed and predicted values of the dependent variable. R^2 is often interpreted as the proportion of the total variation in the dependent variable accounted for by the regression model (5.1). R^2 ranges from 0 to 1. If there is no linear relation between the dependent and independent variables, R^2 is 0 or very small. If all the observations fall on the regression line, R^2 is 1. This measure of the goodness of fit of a linear model is also called the *coefficient of determination*. R_a^2, the adjusted R^2, is designed to compensate for the optimistic bias of R^2. It is a function of R^2 adjusted by the number of variables in the model and the sample size.

$$R_a^2 = R^2 - \frac{p(1-R^2)}{N-p-1} \tag{5.4}$$

where p is the number of independent variables in the equation. The last column in the table, standard error of the estimate, is the square root of the residual mean square in the ANOVA table below. It measures the spread of the residuals (or errors) about the fitted line using the regression model (5.1).

A noticeable point in Table 5.4 is that R^2, the goodness of fitting, has a very high value starting from the first model. It indicates that 90% of the variation of goodput can be explained solely by timeout. Adding SLOWDOWN and RETRANS into the model increases the coefficient of determination, the increase however, is small.

Model	**Predictors**	R	R^2	R_a^2	**Std Error of the Estimate**
1	(Constant), TIMEOUT	.952	.907	.906	4235.416
2	(Constant), TIMEOUT, RETRANS	.956	.914	.913	4077.532
3 3	(Constant), TIMEOUT, RETRANS, SLOWDOWN	.958	.918	.917	3985.823

Table 5.4: Model Summary

Table 5.5 is the analysis of variance or ANOVA table. Denote y_i as the i-th observed value of the dependent variable, $\bar{y}$ as their mean, and $\hat{y}_i$ as the i-th predicted value. The sums of squares for regression, for residual and total are defined as

$$\begin{aligned} \textit{SS Regression} &= \sum_{i=1}^{n}(\hat{y}_i - \bar{y})^2 \\ \textit{SS Residual} &= \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \\ \textit{SS Total} &= \sum_{i=1}^{n}(y_i^2 - \bar{y})^2 \end{aligned} \tag{5.5}$$

The degrees of freedom are listed in the third column. Mean squares are the sums of squares divided by their respective degree of freedom. The F statistic is the ratio of mean square of regression to the mean square of residual. It is used to test the hypothesis that all regression coefficients are zero:

$$b_1 = b_2 = \cdots = b_n = 0 \tag{5.6}$$

i.e., no linear relation exists between the dependent variable and the independent variables. F is large when the independent variables help to explain the variation in the dependent variable. Here the linear relation is highly significant (in all three models, the p value for the F is less than 0.0005).

Model		Sum of Squares	df	Mean Square	F	Sig.
1	Regression	35153253390.678	1	35153253390.678	1959.627	.000
	Residual	3623627402.866	202	17938749.519		
	Total	38776880793.544	203			
2	Regression	35435001755.524	2	17717500877.762	1065.633	.000
	Residual	3341879038.020	201	16626263.871		
	Total	38776880793.544	203			
3	Regression	35599524396.335	3	11866508132.112	746.942	.000
	Residual	3177356397.209	200	15886781.986		
	Total	38776880793.544	203			

Table 5.5: ANOVA

The second column of Table 5.6 lists the estimate of coefficients in the regression model (5.1) to compute the predicted values for the dependent variable. The standard error of the coefficients is listed in the third column. When all variables are transformed into z-score,

$$U = \frac{Y - \bar{Y}}{s_Y}, \quad Z_i = \frac{X_i - \bar{X}_i}{s_{X_i}} \tag{5.7}$$

model (5.1) can be written as

$$U = \beta_1 Z_1 + \beta_2 Z_2 + \cdots + \beta_n Z_n \tag{5.8}$$

with

$$\beta_i = \frac{S_{X_i}}{S_Y} b_i, \quad i = 1, \ldots, n \tag{5.9}$$

where S_{X_i} and S_Y are the standard deviation of X_i and Y. The β's are called standardized coefficients. They are an attempt to make the regression coefficients more comparable. The t statistic in the next column provides some clue regarding the relative importance of each variable in the model. They are obtained by dividing the coefficients by their standard error. Clearly TIMEOUT is much more important than RETRANS and SLOWDOWN. The last column is the significant level calculated from the percentile of the t distribution.

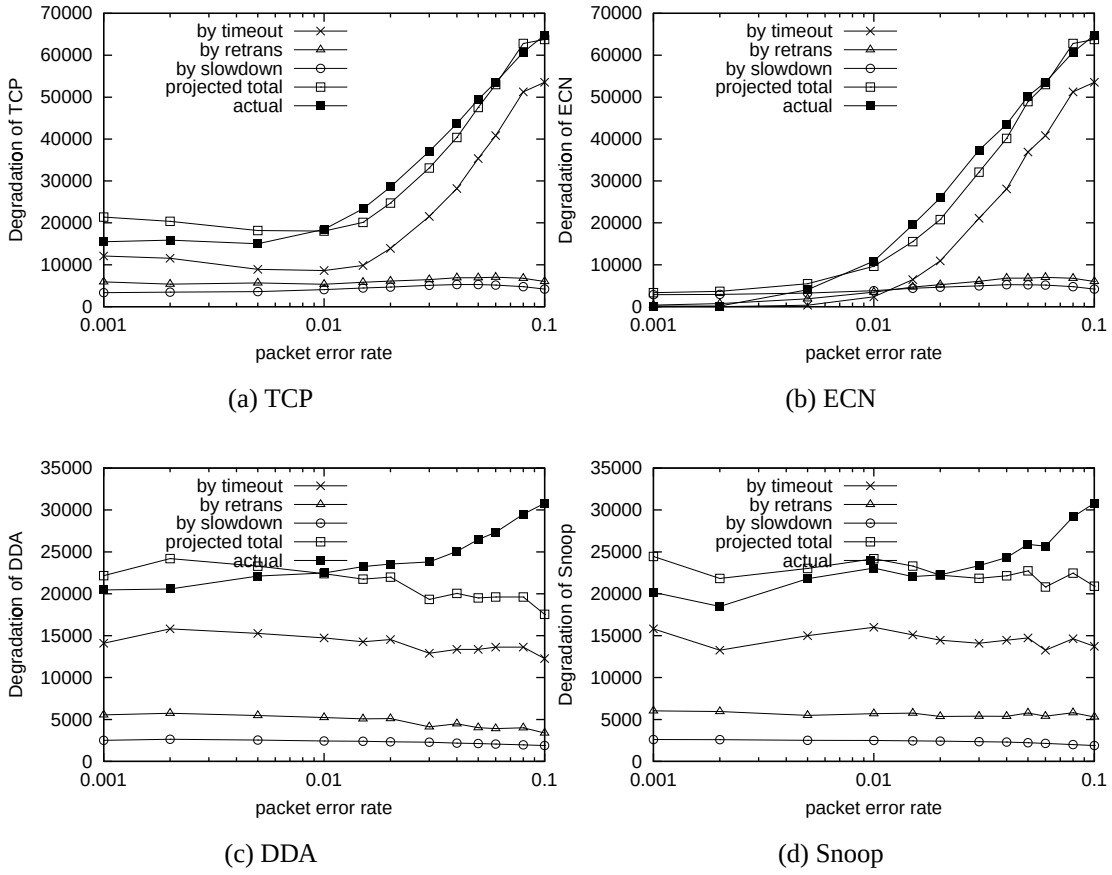
Model		Unstandardized Coefficients		Standardized Coefficients	t	Sig.
		B	Std Error	Beta		
1	(Constant)	68645.314	1490.666		139.902	.000
	TIMEOUT	-103.269	2.333	-.952	-44.268	.000
2	(Constant)	72267.513	998.691		72.362	.000
	TIMEOUT	-94.690	3.064	-.873	-30.906	.000
	RETRANS	-5.078	1.234	-.116	-4.117	.000
3	(Constant)	74879.746	1269.624		58.978	.000
	TIMEOUT	-90.912	3.217	-.838	-28.262	.000
	RETRANS	-5.092	1.206	-.117	-4.223	.000
	SLOWDOWN	-4.341	1.349	-.074	-3.218	.002

Table 5.6: Coefficients

Based on the model consisting of all three variables, performance degradation of each method is broken down according to the three contributors. Figure 5.3 shows the relative size of degradation by the three contributors. The projected total degradation and the actual degradation are also shown. The actual degradation is computed from a hypothetical goodput based on the projection with no retransmission, no slowdown and no timeout.

Obviously timeout makes up the main part of the degradation. Retransmission is the second leading contributor. Contrary to intuition, slowdown has the smallest contribution to degradation.

From Figure 5.2(c)(d) and Figure 5.3(c)(d), we can clearly see that Snoop and DDA have lower performance than plain TCP when the packet error rate is not very high, and the main reason for the poor performance is that they fail to improve the timeout behavior.



(a) TCP (b) ECN

(c) DDA (d) Snoop

Figure 5.3: Breakdown of performance degradation

5.5 Chapter Summary

This chapter analyzes the leading contributors of performance degradation in wireless TCP. Contrary to the common understanding that the inability to distinguish wireless and congestion losses is the main cause of performance degradation, our result indicates that timeout is the leading contributor of TCP performance degradation. It also shows that current enhancements fail to improve the timeout behavior.

The value of this study is that it points out a direction in designing enhancement schemes. Since timeout is the result of multiple losses in one window, a good scheme should stop relying on using packet losses as the mechanism of delivering congestion feedback. Such a scheme will be described in detail in the next chapter.

CHAPTER 6

USING CONGESTION COHERENCE TO ENHANCE TCP OVER WIRELESS LINKS

In the previous two chapters, we have seen that TCP has poor performance over wireless links, and enhancement proposals in the literature do not satisfy the implementation requirements established in Chapter 4. The statistical analysis in Chapter 5 also indicates that timeout as a result of multiple losses in one window is the leading contributor of wireless TCP performance degradation. Current proposals use packet losses to deliver congestion feedback and fail to improve the TCP's timeout behavior over wireless links. A good enhancement scheme must stop using packet losses as the congestion feedback mechanism. In this chapter, we will present such an enhancement scheme.

6.1 Introduction

Transmission Control Protocol (TCP) is the most widely used transport protocol in the network world. By providing reliable data transport and congestion control, TCP serves as the foundation of today's Internet. Historically, TCP was designed mainly for wired networks where transmission errors are rare and the majority of packet losses are caused by congestion. Jain was the first to point out that packet loss and the resulting timeout at the

source are indications of congestion and the sources should reduce their traffic on timeout [41].

With the rapid development of wireless networks in recent years, TCP has been deployed on wireless networks. It has been discovered that TCP has poor performance over wireless links. This is because the assumption behind TCP congestion control algorithm — that the majority of packet losses are caused by congestion — is no longer true on the wireless links. When a wireless loss is treated as a congestion loss, the effective TCP transmission rate is cut by half. If wireless errors happen frequently, the effective transmission rate of the wireless link becomes almost zero.

A scenario of such transmission rate drop is illustrated in Figure 6.1. Before the transmission error happens, the path between the sender and the receiver can support a window size of six packets. The sender transmits at this rate and a packet is lost on the wireless link due to a transmission error. When the destination sees the out-of-order packets, it follows TCP congestion control algorithm to send back duplicate ACKs. Upon the receipt of three duplicate ACKs, the sender assumes that congestion has happened in the network, retransmits the lost packet and reduces the window by half. As a result, the wireless link that can send six packets in a window is now sending only three, even though the network is not congested.

In the past few years, a number of enhancements have been proposed to improve TCP performance. These enhancements suggest modification at the sender, the base station or the receiver. Some of them try to hide the lossy characteristic of wireless link from TCP by performing buffering and retransmission at the base station. Others use extra traffic to determine the network congestion status. These methods are reviewed and evaluated in the next section of this chapter. It is our conclusion that these methods either make unrealistic

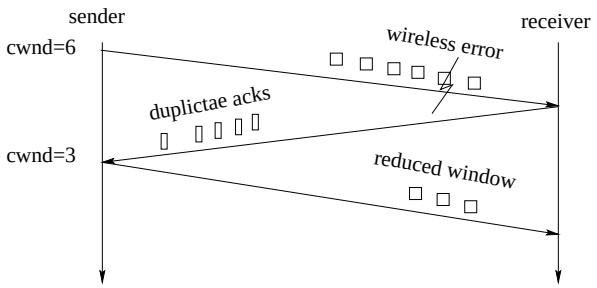


Figure 6.1: Window reduction due to wireless losses

assumptions, do not meet certain implementation requirements, or apply only to specific network configurations.

In this chapter, we identify the three ways that wireless errors degrade TCP performance, and propose a new enhancement approach. This new approach makes use of the sequential coherence of congestion marks to distinguish wireless losses from congestion losses. It requires only minor modifications in the mobile host, and applies to all network configurations. Analysis and simulation results show that this approach avoids the majority of retransmissions, unnecessary slowdown and timeouts, and thus significantly improves TCP performance.

6.2 Proposed Approach

As discussed above, transmission errors cause the poor TCP performance over wireless links. A closer look at TCP traces reveals that wireless errors can degrade TCP performance in three ways:

End-to-end retransmissions: When a packet is lost in the wireless link, the packet has to be retransmitted. If the retransmission is done end-to-end, the extra traffic can

add to the congestion and reduce the number of packets that can be sent through the bottleneck link. Extra traffic also causes a longer delay.

Unnecessary slowdowns: When wireless losses are retransmitted end-to-end, the regular TCP algorithm treats the packet loss as an indication of congestion and thus reduces the congestion window size. Unnecessary slowdowns cause significant performance degradation because the effective transmission rate is cut to half.

Timeouts: The existing TCP algorithms (Tahoe, Reno or new Reno) can recover one packet loss in a window using duplicate acknowledgments. Two or more packet losses (can be congestion or wireless) in the same window will result in a timeout. Timeouts cause severe performance degradation because it takes many round trip times (RTT) to bring the transmission rate to a normal value after a timeout. If packet loss is used for congestion control as in the current TCP algorithm, then any wireless loss within the RTT that contains a congestion loss always results in a timeout.

In order to improve TCP performance, an enhancement solution should eliminate the above three degradations. The first degradation can be eliminated by local link layer retransmission. The second degradation can be eliminated if a mechanism to distinguish between wireless and congestion losses is found. In order to eliminate the third degradation, we need to stop using congestion loss as a mechanism to adjust the window.

We find that all current proposals eliminate only the first two degradations. They all use packet losses to deliver congestion feedback. Therefore, the third degradation is inherent and unavoidable.

In this chapter, we use Explicit Congestion Notification (ECN) for congestion control and propose a mechanism to distinguish wireless and congestion losses. ECN avoids congestion losses by making use of early congestion warnings. Avoiding congestion losses has two benefits on performance. It avoids the end-to-end retransmissions, as well as timeouts caused by multiple losses in a window. In this way, we can eliminate all the three degradations of TCP performance.

Our proposal consists of the following three parts:

- *Assume that all network routers are ECN capable.*
- *Implement local link layer retransmission on the wireless link to avoid end-to-end retransmission of the wireless losses,*
- *Use congestion coherence to determine cause of packet loss at the mobile host.*

Details of the proposed approach are given below.

6.2.1 Explicit Congestion Notification

The idea of marking packet header at the congested router to deliver binary feedback on congestion status can be traced to the *DECbit* scheme [67] in 1988. Compared with timeout and duplicate ACKs, explicit feedback delivers a direct and faster congestion signal to the sender.

Explicit Congestion Notification, the binary feedback scheme in TCP/IP, was first introduced by Floyd and Ramakrishnan [26]. ECN uses two bits in the IP header and two bits in the TCP header for ECN capability negotiation and feedback delivery. When the queue length exceeds a threshold and the incoming packet is labeled ECN-Capable, the

router marks the packet as *congestion experienced*. At the destination, the Congestion Experience bit is copied to the *ECN-Echo* bit in the TCP acknowledgment and thus delivers a congestion notification back to the sender. Upon receiving the ECN-Echo, the sender reduces its congestion window to alleviate the congestion.

ECN has proven to be effective in avoiding unnecessary packet drops and in improving TCP performance. In RFC 2309 [24], it was recommended to be widely deployed as a router mechanism on the Internet, and was standardized by IETF in RFC 2481 [66] in 1999. In this paper, we will assume ECN is implemented in all routers. This is a very important assumption of our proposal.

The marking policy determines how a packet is marked. Floyd and Ramakrishnan [26] recommend using Random Early Detection (RED) [31] as the marking policy. RED detects incipient congestion and randomly marks packets at a computed probability when the average queue size exceeds a threshold. Figure 6.2 shows the marking policy used in our simulation, which is a little different from that in [26]. We use the *actual* queue size instead of the average queue size. When the queue size is below a minimum threshold th_{min}, the incoming packet will not be marked. When the queue size is between th_{min} and a maximum threshold th_{max}, the incoming packet will be marked with a probability proportional to the queue size. When the queue size is greater than th_{max}, all incoming packets are marked. When the queue size is equal to buffer size, all incoming packets are dropped.

6.2.2 Local Link Layer Retransmission

When a packet is lost in the wireless media, it has to be retransmitted, either end-to-end or over the local wireless link. Compared to the end-to-end retransmission, local link layer

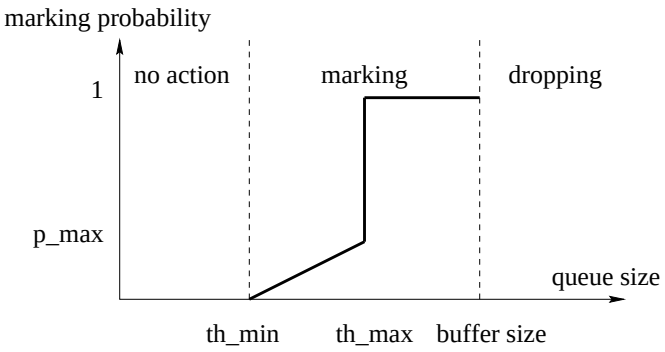


Figure 6.2: Marking and dropping policy used for congestion coherence

retransmission not only avoids extra traffic in the wired network but also reduces the delay. However, the delay caused by the local link layer retransmission can interfere with the end-to-end TCP retransmission timer. If the local link layer retransmission takes too long, the source may timeout and trigger an end-to-end retransmission before the acknowledgment of local retransmission is received. In designing the link layer retransmission scheme, it is important to transmit the failed packets with high priority to reduce the retransmission delay. One way of implementing this is to use the "insert from front" strategy. When a packet is detected to be lost, the link layer inserts the failed packet into the front of the transmission queue and transmits it when the media is available. In this way, the failed packet does not suffer the queuing delay again.

An implication of local link layer retransmission is out-of-order packets. When a packet is lost and retransmitted, the TCP receiver will receive out-of-order packets and may respond with duplicate ACKs. Both congestion losses and wireless losses cause out-of-order packets and create holes in the packet sequence number space, but their consequences are different. A hole caused by a wireless loss will be filled when the retransmission arrives,

but a hole caused by a congestion loss will not be filled without a timeout or triggering an end-to-end retransmission. If the receiver knows the hole is a wireless loss, it should wait for the retransmission. If it knows the hole is a congestion loss, it should trigger the end-to-end retransmission right away.

The receiver needs a scheme to tell whether hole is a wireless loss or a congestion loss. Such a scheme is described in the next section.

6.2.3 Congestion Coherence

Table 6.1 illustrates two loss cases taken from a simulation trace. Packets 37 and 112 are lost, but the ECN marks of their neighboring packets are listed in the table. An "E" means the packet is marked "Congestion Experienced", a blank means it did not experience any congestion. By looking at these markings, is it possible to guess which packet is lost due to congestion and which is lost due to error?

seqno	mark
33	
34	E
35	E
36	E
37	lost
38	E
39	E
40	

seqno	mark
108	
109	
110	
111	
112	lost
113	
114	
115	

Table 6.1: Are these packets lost due to congestion or due to transmission error?

Packet 37 is a congestion loss and packet 112 is a transmission loss. This observation is based on the so called *congestion coherence* of ECN markings. It reflects the fact that

congestion does not happen or disappear suddenly. Before congestion becomes so severe that a packet has to be dropped, some packets must have been marked. Similarly, after a packet is dropped, congestion does not disappear immediately. The queue size falls gradually and some packets are marked. As a result, congestion losses are normally preceded and followed by marked packets, see Figure 6.3.

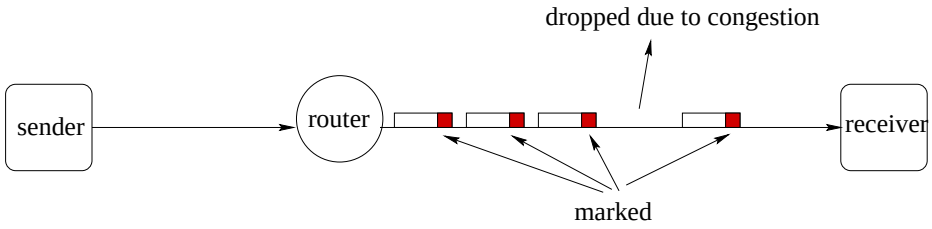


Figure 6.3: Congestion coherence

In contrast, transmission errors normally happen independent of congestion. Neighboring packets of a wireless loss are usually not marked.

Therefore, being surrounded by marked packets is a tag of congestion losses. Congestion coherence can be used to distinguish congestion losses from wireless losses. We define the *coherence context* of packet n as packets $\{n-1, n+1, n+2\}$. The coherence context is said to be marked if any packet in the context is marked.

If a packet is found lost at the destination and its context is marked, duplicate acknowledgments should be sent right away to invoke fast retransmit and congestion control at the source. If the context is not marked, it is most likely a wireless loss. Duplicate acknowledgments should be held to avoid invoking fast retransmit and congestion control at the source.

The same idea can be applied to the wireless sender case. When the wireless sender receives duplicate acknowledgments, it checks whether the coherence context contains an ECN-Echo. If yes, then the duplicate acknowledgments are most likely caused by a congestion loss, so the sender invokes the congestion control. Otherwise, the duplicate acknowledgments are most likely caused by a wireless loss. The sender ignores duplicate acknowledgments until the local retransmission succeeds.

6.2.4 Congestion Coherence TCP Algorithm

- The TCP sink follows existing algorithm for sending new acknowledgments, first and second duplicate acknowledgments.
- When the third duplicate acknowledgment is to be sent, TCP sink checks whether the coherence context is marked. If yes, the acknowledgment is sent right away. Otherwise, it is deferred for w seconds, and a timer is started.
- If the expected packet arrives during the w seconds, a new acknowledgment is generated and the timer is cleared.
- If the timer expires, all deferred duplicate acknowledgments are released.

Figure 6.4: Congestion Coherence receiving algorithm

In our proposed approach, the modifications to existing TCP algorithms are made in the wireless end. Based on the technique discussed above, this approach is named *Congestion Coherence*. Figures 6.4 and 6.5 show the modified receiving and sending algorithms.

It should be noticed that the modifications to the receiving and sending algorithms are made on the same end. The implementation of Congestion Coherence at the wireless end hides the lossy characteristic of the wireless link from the other end. The wired end and intermediate routers, including the base station, need no modification. If the wireless link is in the middle, such as a satellite link or in an ad-hoc wireless network, the modifications can be made on either end or both ends.

Revisiting Table 4.2, we can notice that our approach meets all the requirements. The traffic overhead is zero because feedback is delivered using the IP and TCP headers and no extra packets are sent. The buffering is minimal, only those packets sent over the wireless link but not yet acknowledged are buffered. The computing complexity is very small, only a few lines of modification need to be added to the existing TCP code.

6.3 Mistake Scenarios

Congestion coherence provides an educated guess of the cause of packet losses. However, this method is based on probability and can make mistake in some cases. In this section, we discuss these mistake cases and analyze their performance impact.

The first mistake case happens when a wireless error occurs in a congestion episode. Since neighboring packets are marked, the wireless loss may be treated as a congestion loss. Therefore, congestion control is triggered at the source and an end-to-end retransmission is started. In this case, the end-to-end retransmission is unnecessary because local link layer

- The TCP sender follows existing algorithm for sending packets and updating the congestion window upon receiving new acknowledgments, first and second duplicate acknowledgments.
- When the third duplicate acknowledgment arrives, the sender checks whether any acknowledgment in the coherence context is an ECN-Echo. If yes, the packet corresponding to the duplicate acknowledgments is sent right away and congestion window is reduced to half if a reduction has not been done in the previous RTT. Otherwise, the sender ignores the duplicate acknowledgement and a timer of w seconds is started.
- If a new acknowledgment arrives during the w seconds, the timer is cleared and new packets are sent as if the duplicate acknowledgments did not happen.
- If the timer expires, the packet corresponding to the duplicate acknowledgments is sent and congestion window is reduced to half if a reduction has not been done in the previous RTT.

Figure 6.5: Congestion Coherence sending algorithm

retransmission will deliver the packet again. However, triggering the congestion control is needed because of the existence of congestion. As congestion control is more important than retransmitting a single packet, sending the third duplicate acknowledgment right away is the correct action.

The second mistake case happens when a burst of background traffic causes a congestion loss without marking the coherence context. In this case, our method regards the packet loss as a wireless loss and will wait w seconds. At the end of this period, the retransmission of this packet does not come. So deferred duplicate ACKs are released to trigger an end-to-end retransmission. The cost of this mistake is a delay in the end-to-end retransmission and a slowdown.

These two mistakes affect the performance, but results from our simulations show that the mistake rate is very small and the impact is minimal.

6.4 Simulation and Result Analysis

In order to compare our proposed method with other proposals, we performed a set of simulations with the ns simulator [53]. In this section, we present our network model, simulation scenarios, data collection method and results from analyzing individual simulation traces and aggregate measurements.

6.4.1 Simulation Model and Scenario Design

The simulations are performed on the simplified network model shown in Figure 5.1, where s_1, s_2 are the sources and d_1, d_2 are the destinations. The link between intermediate routers r_1 and r_2 is the bottleneck link. The link between r_2 and d_1 is a wireless link. The numbers beside each link represent its rate and delay.

The experiment traffic is an FTP session from s_1 to d_1 using TCP Reno as the transport protocol. The background traffic is a UDP flow from s_2 to d_2 generated by an exponential on-off model. The mean burst period and the mean silence period are 100 ms, and the burst data rate is 500 kbps. Both TCP and UDP packet sizes are 1000 bytes, and TCP acknowledgments are 40 bytes long.

Link layer retransmission is implemented on the wireless link. Packets sent but not acknowledged at the link level in 40 ms are resent. Retransmitted packets are also subject to wireless errors at the same rate. The waiting time at the receiver is set at 81 ms so that packets delivered within two retransmissions are accepted. The packet error rate of the wireless link is varied to test the performance of various proposals under different loss scenarios.

To reflect steady state measurement, the simulation duration should be long enough to minimize the effect of the initial transient state. Longer simulation normally generates smoother aggregate results. Techniques to determine the simulation end time can be found in Chapter 25 of [42]. In our experiment, we tried various durations and found the results of 500 seconds show the essential features without noticeable difference from longer simulations, so all aggregate measurements are collected from 500-second simulations.

The proposals we compared include base TCP, DDA, Snoop and Congestion Coherence. When the mobile host is a sender, Snoop is replaced by ELN. In order to show that ECN without congestion coherence does not work, we also compared against plain ECN. We did not compare against I-TCP, Multiple Acknowledgments and Control Connection because they violate the end-to-end and the local requirements and are considered unacceptable.

We experimented with various network configurations, including the wireless link as the last hop (mobile receiver), as the first hop (mobile sender) and as intermediate links (e.g. ad-hoc or satellite link). Congestion Coherence works for all three configurations and has similar performance. Because most other proposals only work for the mobile receiver configuration, the comparison below is conducted only for this configuration.

6.4.2 Results and Analysis

Our first group of results, shown in Figure 6.6, is the TCP congestion window and queue length of each proposal. They are collected from 40-second simulation traces. The packet error rate in the simulation is 0.1. A calculation shows that the average window size of the wireless connection can be roughly 10 packets, but as shown in the figure, the window size of base TCP and ECN is reduced frequently. Their corresponding queue size graph shows the queue at the bottleneck link is almost always empty. Therefore, their link efficiency is very low. Snoop and DDA solve the problem of unnecessary slowdowns caused by wireless errors. The window size is significantly increased and the bottleneck link is better utilized. Nevertheless, the spikes in the bottom of Snoop and DDA *cwnd* figure indicate these two methods suffer severe degradation from timeouts. Congestion Coherence is a thorough solution. Unnecessary slowdowns and timeouts are avoided. The queue size figure shows it has high link efficiency.

The major metric to evaluate the enhancement proposals is *goodput*, which is defined as the number of packets successfully received and acknowledged by the mobile host, excluding the retransmitted packets. The goodput of the five proposals under different packet error rates is shown in Figure 6.7. Base TCP performs reasonably well when the packet error rate is very small, but as the packet error rate increases, its performance degrades

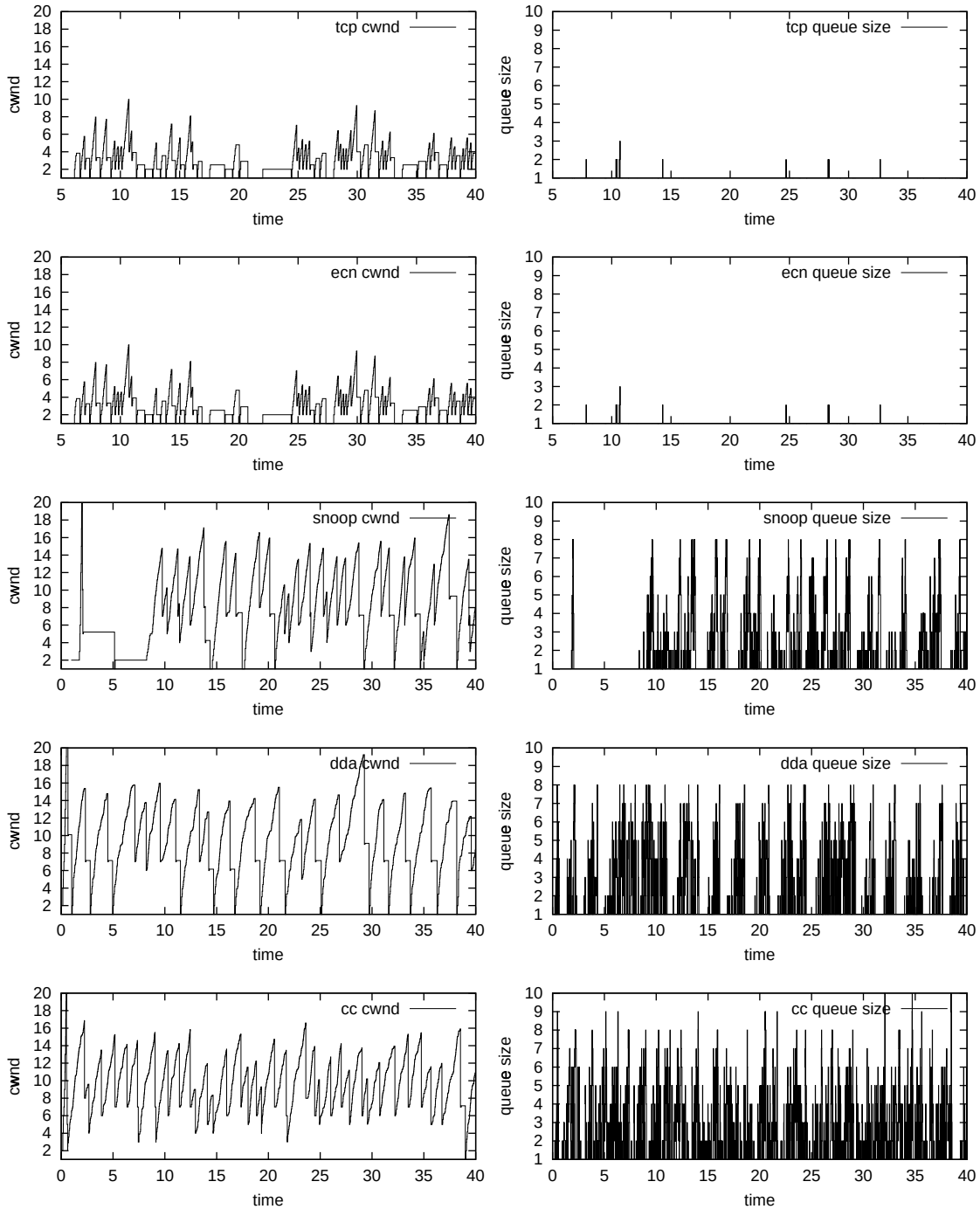


Figure 6.6: Congestion window and queue length for different methods

quickly. The performance curve confirms that TCP needs enhancement on wireless links. Plain ECN performs better than base TCP when the error rate is very small, but its performance degrades quickly as the error rate increases. DDA does not degrade much with the error rate, but its performance under small error rates is low. Congestion Coherence improves goodput for all packet error rates in the simulated range.

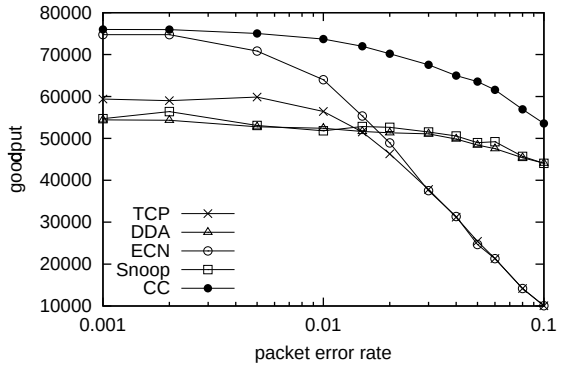


Figure 6.7: Goodput for the five methods

In addition to the goodput, we also analyzed the simulation trace and collected other data that helped us understand why one enhancement works better than another.

Figure 6.8 and 6.9 show the number of wireless and congestion losses. The number of wireless losses equals the total number of packets transmitted on the wireless link times the packet error rate. The number of congestion losses of base TCP, Snoop and DDA is significantly more than other methods because they use packet losses as a congestion control mechanism. As the packet error rate increases, wireless losses reduce the congestion window so frequently that the window seldom grows to the level that a packet needs to

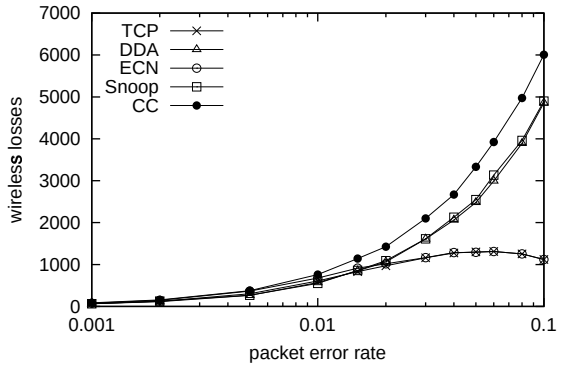


Figure 6.8: Wireless losses

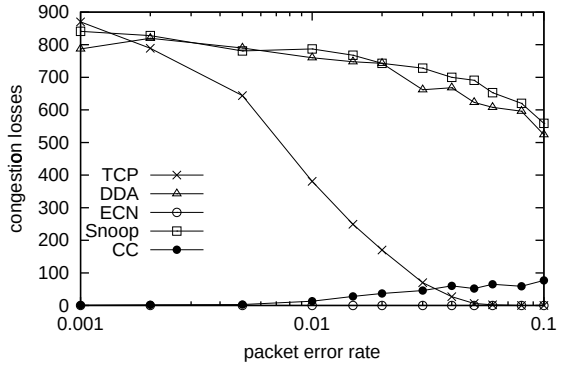


Figure 6.9: Congestion losses

be dropped. This explains the smaller number of congestion losses of base TCP in the right half of Figure 6.9. In contrast, methods using ECN do not suffer from congestion losses on a regular basis. Congestion losses happen only when bursts of background traffic generate so many packets that the buffer of bottleneck link cannot absorb. As analyzed in the beginning of Section 6.2, having fewer congestion losses helps to reduce end-to-end retransmissions and the chance of timeout.

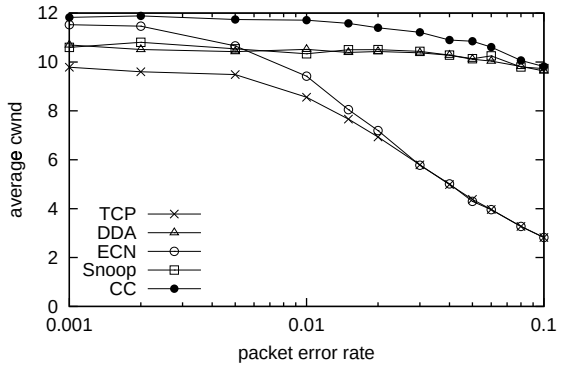


Figure 6.10: Average congestion window size

Figure 6.10 shows the average congestion window size. As the packet error rate increases, wireless losses cause unnecessary slowdowns in TCP and plain ECN, but the window size of Snoop, DDA and Congestion Coherence is not affected much by wireless errors. The slight drop in the right upper corner is caused by transmission errors in the retransmit packets[3]. This figure confirms that Snoop, DDA and Congestion Coherence solve the problem of unnecessary slowdowns.

Figure 6.11 shows the number of timeouts that occurred during the simulation period. When the packet error rate is small, TCP, Snoop and DDA have the largest number of timeouts because they use packet losses for congestion control. Their buffer occupancy at the bottleneck link can grow so high that bursts in background traffic can cause continual losses. Since two or more losses in a window cause a timeout, this translates to large number of timeouts. ECN and Congestion Coherence have very few timeouts because most of their congestion losses are avoided; background traffic causes occasional losses, but seldom become multiple losses in one window. As the packet error rate increases, the

[3]In another simulation, when the retransmission method is replaced by perfect retransmission, i.e., no transmission error for retransmitted packet, wireless losses do not affect the congestion window at all.

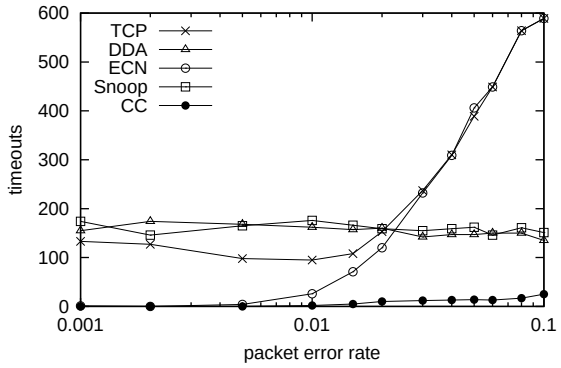


Figure 6.11: Number of timeouts

number of timeouts in TCP and plain ECN increases dramatically because larger number of wireless losses increases the chance of multiple losses in one window. When the error rate is below 0.014, TCP has more timeouts than plain ECN. As the congestion window of TCP is reduced frequently by wireless losses (Figure 6.10) and congestion losses become fewer (Figure 6.9), TCP behaves almost identical to ECN. The timeouts of Snoop and DDA are mainly caused by congestion losses and remain constant across all packet error rates. Congestion Coherence has the smallest number of timeouts of all proposals. This figure is the evidence showing that only our proposed scheme avoids the degradation caused by timeouts.

Figure 6.12 shows the number of end-to-end retransmissions. This number depends on the number of congestion losses, wireless losses and timeouts, as well as the enhancement method used. In fact, congestion losses in all methods are retransmitted. Wireless losses in TCP and plain ECN are retransmitted. When timeout happens, one full window of packets are retransmitted. Snoop and DDA avoid the majority of end-to-end retransmissions of wireless losses, but they still have a large number of retransmissions because of congestion

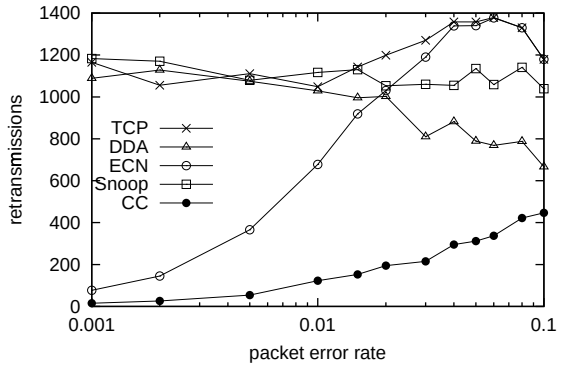


Figure 6.12: Number of end-to-end retransmissions

losses and timeouts. Plain ECN reduces congestion losses, but cannot recover from wireless losses. All its wireless losses are retransmitted. Congestion Coherence avoids the majority of congestion losses, and waits for the local retransmission of wireless losses, so it has the smallest number of retransmissions.

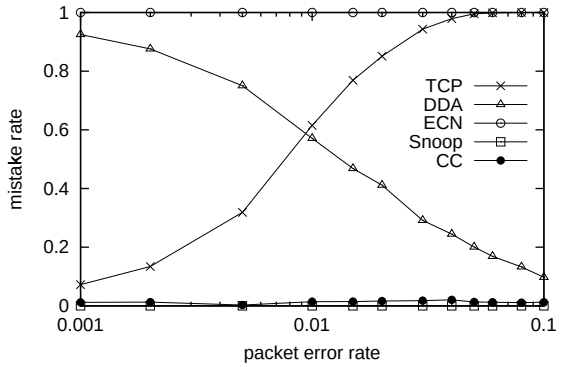


Figure 6.13: Mistake rate

Finally, the mistake rate in determining the cause of packet losses is shown in Figure 6.13. Both base TCP and plain ECN assume all losses are caused by congestion, so their mistake rate is the percentage of wireless losses in all losses. Plain ECN makes almost the same number of mistakes as base TCP, but it has a much higher mistake rate because of its small number of congestion losses. DDA assumes all packet losses are due to wireless errors, so its mistake rate decreases when the packet error rate increases. Snoop knows the exact cause of all packet losses by monitoring packets arriving at the base station, so its mistake rate is zero. Congestion Coherence takes advantage of congestion coherence and makes the right guess in most cases. As analyzed in Section 6.3, it makes mistake when very bursty traffic causes sudden packet losses without having neighboring packets marked. In our simulations, Congestion Coherence's mistake rate ranges from 0.06% to 1.2%. This rate is very small compared with other methods (except Snoop), and has minimal impact on the performance.

In summary, the simulation results reveal that Congestion Coherence avoids the majority of congestion losses and is able to distinguish wireless loss from congestion losses. It is the only enhancement that avoids the three degradations of TCP performance over wireless links — end-to-end retransmissions, unnecessary slowdowns and timeouts. Therefore, the performance of TCP is significantly improved.

6.5 Discussions

This section is devoted to discussing a number of implementation details, alternatives and possible extensions for future study.

Coherence Context The coherence context used in our simulation is three packets, one before the lost packet and two after. We tried different sizes of the coherence context. It turned out that smaller coherence contexts tend to mistake congestion losses as wireless losses and cause more timeouts while larger coherence contexts tend to mistake wireless losses as congestion losses and cause more unnecessary end-to-end retransmissions. The other coherence context of size three, i.e., two before the lost packet and one after, yields similar results.

Location of Wireless Link Most enhancement proposals in the literature assume the wireless link is the last hop, congestion losses happen only between the fixed host and the base station, and wireless errors happen only between the base station and the mobile host. When the wireless link is a hop that connects two wired networks, like the satellite links, or when there are multiple wireless links as in an ad-hoc network, this assumption is no longer true. These solutions do not work in these cases. Our solution does not assume the location or the number of wireless links. As long as ECN is used in intermediate routers and the wireless links implement local retransmission, our solution will work.

Mark-Front Strategy It should be noted that the mark on packets carries the congestion information of the path to the destination. The earlier the information is delivered to the sender, the more effective the sender's response can be. In our recent paper [55], we proposed to use the packet at the front of the queue, instead of the packet at the end of the queue, to carry the congestion information. This is called the mark-front strategy and has been shown to require smaller buffers, to generate higher throughput and to provide better fairness. In this chapter, we assume marking and dropping are always performed on the

packet in the front of the queue. Marking is done when a packet is leaving the queue and dropping is done when a packet enters a full queue.

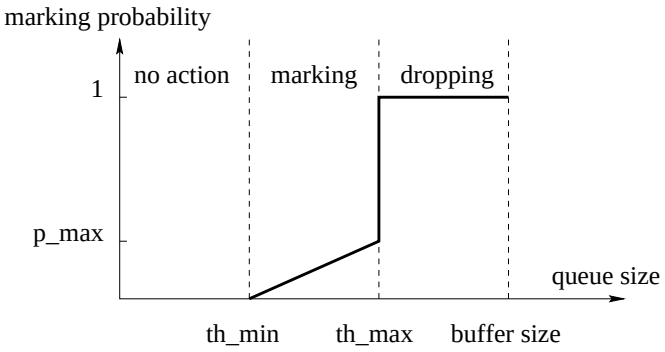


Figure 6.14: Early RED marking and dropping policy

Marking Policy The marking policy has a significant impact on congestion coherence. It is important to ensure that packets are not dropped without neighboring packets being marked. The randomness in deciding whether to mark a packet helps to desynchronize the TCP congestion windows among flows, but may jeopardize this assurance. Early versions of the RED algorithm, as depicted in Figure 6.14, yield low congestion coherence. The *gentle* option described in [29] and implemented in *ns2.17* improves the coherence. A better marking policy should keep a random marking zone to help desynchronize TCP congestion windows among flows, and a deterministic marking zone to ensure congestion coherence, as shown in Figure 6.2.

Queue Length It should be noted that it is the real queue length that is related to packet losses. Average queue length, because of its delay in reflecting the real congestion status,

normally yields lower coherence and low goodput. Our simulations confirm this result. Discussing options of setting the RED parameters is beyond the scope of this dissertation. We will report them elsewhere.

6.6 Chapter Summary

This chapter presents a scheme called Congestion Coherence to enhance TCP over wireless links. Based on the analysis on the leading contributors of wireless TCP performance degradation presented in 5, this scheme stops relying on using packet losses as the mechanism to deliver congestion feedback to the sender. It uses ECN to deliver the congestion feedback and uses the "congestion coherence" in consecutive packets to determine the cause of packet losses.

In terms of performance, the proposed scheme avoids the majority of end-to-end retransmissions, unnecessary slowdowns and timeouts caused by wireless errors, and therefore improves the performance of TCP over wireless links.

In terms of modifications, the proposed scheme requires only minor modification in the TCP code at the mobile station's side. No modification is needed in the base station and in the fixed host, assuming ECN has been implemented in all network routers. In this way, TCP's end-to-end semantic is maintained; all modifications are in the scope of wireless service providers; the scheme can work with encrypted traffic and applies to two-way traffic. In addition, this scheme applies to immediate wireless links, such as satellite links and mobile networks.

These key differentiators establish the scheme as a unique wireless TCP enhancement. The proposed enhancement clearly shows the advantages of ECN over traditional congestion control mechanisms. Congestion Coherence is highly dependent on the wide deployment of ECN protocol. We recommend that it be used when ECN is widely deployed.

In conclusion, we regard this scheme as a new wireless TCP enhancement as well as a promotion of the ECN protocol.

CHAPTER 7

IMPROVING EXPLICIT CONGESTION NOTIFICATION WITH THE MARK-FRONT STRATEGY

Fast congestion feedback from the network is crucial to the effectiveness of TCP congestion control actions. Current TCP/IP networks use packet losses to signal congestion. Packet losses not only reduce TCP performance, but also introduce large delays. Explicit Congestion Notification (ECN) delivers a faster indication of congestion and improves performance. However, current ECN implementations mark the packet from the tail of the queue. In this chapter, we propose a new strategy that delivers even faster congestion feedback.

7.1 Introduction

Delivering congestion signals is essential to the performance of computer networks. In TCP/IP, congestion signals from the network are used by the source to determine the load. When a packet is acknowledged, the source increases its window size. When a congestion signal is received, its window size is reduced [39, 71].

TCP/IP uses two methods to deliver congestion signals. The first method is timeout. When the source sends a packet, it starts a retransmission timer. If it does not receive an acknowledgment within a certain time, it assumes congestion has occurred in the network

and the packet has been lost. Timeout is the slowest congestion signal because the source has to wait a long time for the retransmission timer to expire.

The second method is loss detection. In this method, the receiver sends a duplicate ACK immediately on reception of each out-of-sequence packet. The source interprets the reception of three duplicate acknowledgments as a congestion packet loss. Loss detection can avoid the long delays of timeouts.

Both timeout and loss detection use packet losses as congestion signals. Packet losses not only increase the traffic in the network, but also add a large transfer delay. The Explicit Congestion Notification (ECN) proposed in [65, 26] provides a light-weight mechanism for routers to send a direct indication of congestion to the source. It makes use of two experimental bits in the IP header and two experimental bits in the TCP header. When the average queue length exceeds a threshold, the incoming packet is marked as *congestion experienced* with a probability calculated from the average queue length. When the marked packet is received, the receiver marks the acknowledgment using an *ECN-Echo* bit in the TCP header to send congestion notification back to the source. Upon receiving the ECN-Echo, the source halves its congestion window to help alleviate the congestion.

Many authors have pointed out that marking provides more information about the congestion state than packet dropping [69, 32], and ECN has been proven to be a better way to deliver the congestion signal and exhibits a better performance [26, 69, 16].

In most ECN implementations, when congestion happens, the congested router marks the incoming packet that just entered the queue. When the buffer is full or when a packet needs to be dropped as in Random Early Detection (RED), some implementations, such as the *ns* simulator [53], have the “drop from front” option as suggested by Yin [76] and Lakshman [50]. A brief discussion of drop from front in RED can be found in [28]. However,

for packet marking, these implementations still mark the incoming packet and not the front packet. We call this policy "mark-tail".

In this chapter, we propose a simple marking mechanism — the "mark-front" strategy. This strategy marks a packet when the packet is going to leave the queue and the queue length is greater than the pre-determined threshold. The mark-front strategy is different from the current mark-tail policy in two ways. First, since the router marks the packet at the time when it is sent, and not at the time when the packet is received, a more up-to-date congestion signal is carried by the marked packet. Second, since the router marks the packet in the front of the queue and not the incoming packet, congestion signals do not undergo the queueing delay as the data packets do. In this way, a faster congestion feedback is delivered to the source.

The implementation of this strategy is extremely simple. One only needs to move the marking action from the enqueue procedure to the dequeue procedure and choose the packet leaving the queue in stead of the packet entering the queue.

We justify the mark-front strategy by studying its benefits. We find that, by providing faster congestion signals, the mark-front strategy reduces the buffer size requirement at the routers, it avoids packet losses, and it improves the link efficiency when the buffer size in routers is limited. Our simulations also show that the mark-front strategy improves the fairness among old and new connections, and alleviates TCP's discrimination against connections with large round trip time.

The mark-front strategy differs from the "drop from front" option in that when packets are dropped, only implicit congestion feedback can be inferred from timeout or duplicate ACKs. When packets are marked, explicit and faster congestion feedback is delivered to the source.

Gibbons and Kelly [32] suggested a number of mechanisms for packet marking, such as "marking all the packets in the queue at the time of a packet loss", "marking every packet leaving the queue from the time of a packet loss until the queue becomes empty", and "marking packets randomly as they leave the queue with a probability so that later packets will not be lost." Our mark-front strategy differs from these marking mechanisms in that it is a simple marking rule that faithfully reflects the up-to-date congestion status, while the mechanisms suggested by Gibbons and Kelly either do not reflect the correct congestion status, or need sophisticated probability calculations for which no sound algorithm is known.

It is worth mentioning that the mark-front strategy is as effective in high-speed networks as in low speed networks. Lakshman and Madhow [49] showed that the amount of drop-tail switches should be at least two to three times the bandwidth-delay product of the network in order for TCP to achieve decent performance and to avoid losses in the slow start phase. Our analysis in section 4.3 reveals that in the steady-state congestion avoidance phase, the queue size fluctuates from empty to one bandwidth-delay product. So the queueing delay experienced by packets when congestion happens is comparable to the fixed round-trip time[4]. Therefore, the mark-front strategy can save as much as a fixed round-trip time in the congestion signal delay, independent of the link speed.

We should also mention that the mark-front strategy applies to both wired and wireless networks. When the router threshold is properly set, the coherence between consecutive packets can be used to distinguish packet losses due to wireless transmission error from packet losses due to congestion. This result will be reported elsewhere.

[4]The fixed round-trip time is the round-trip time under light load, i.e., without the queueing delay.

This chapter is organized as follows. In section 2 we describe the assumptions for our analysis. Dynamics of queue growth with TCP window control is studied in section 3. In section 4, we compare the buffer size requirements of mark-front and mark-tail strategies. In section 5, we explain why mark-front is fairer than mark-tail. The simulation results that verify our conclusions are presented in section 6. In section 7, we remove the assumptions made to facilitate the analysis, and apply the mark-front strategy to the RED algorithm. Simulation results show that mark-front has the advantage over mark-tail as shown by the analysis.

7.2 Assumptions

ECN is used together with TCP congestion control mechanisms like slow start and congestion avoidance [71]. When the acknowledgment is not marked, the source follows existing TCP algorithms to send data and increase the congestion window. Upon the receipt of an ECN-Echo, the source halves its congestion window and reduces the slow start threshold. In the case of a packet loss, the source follows the TCP algorithm to reduce the window and retransmit the lost packet.

ECN delivers congestion signals by setting the *congestion experienced* bit, but determining when to set the bit depends on the congestion detection policy. In [65], ECN is proposed for use with average queue length and RED. Their goal is to avoid sending congestion signals caused by transient traffic and to desynchronize sender windows [31, 11]. In this chapter, to allow analytical modeling, we assume a simplified congestion detection criterion: when the *actual queue length* is smaller than the threshold, the incoming packet will not be marked; when the *actual queue length* exceeds the threshold, the incoming packet will be marked.

We also make the following assumptions: (1) Receiver windows are large enough so the bottleneck is in the network. (2) Senders always have data to send and will send as many packets as their windows allow. (3) There is only one bottleneck link that causes queue buildup. (4) Receivers acknowledge every packet received and there are no delayed acknowledgments. (5) There is no ACK compression [78]. (6) The queue length is measured in packets and all packets have the same size.

7.3 Queue Dynamics with TCP Window Control

In this section, we study the relationship between the window size at the source and the queue size at the congested router. The purpose is to show the difference between mark-tail and mark-front strategies. Our analysis is made on one connection, but with small modifications, it can also apply to multiple connection cases. Simulation results of multiple connections and connections with different round trip times will be presented in section 6.

In a path with one connection, the only bottleneck is the first link with the lowest rate in the entire route. In case of congestion, the queue builds ups only at the router before the bottleneck link. The following lemma is obvious.

Lemma 1 *If the data rate of the bottleneck link is d packets per second, then the downstream packet inter-arrival time and the ack inter-arrival time on the reverse link can not be shorter than $1/d$ seconds. If the bottleneck link is fully-loaded (i.e., no idling), then the downstream packet inter-arrival time and the ACK inter-arrival time on the reverse link are $1/d$ seconds.*

Denoting the source window size at time t as $w(t)$, we have

Theorem 1 *Consider a path with only one connection and only one bottleneck link. Let the fixed round trip time be r seconds, the bottleneck link rate be d packets per second, and the propagation and transmission time between the source and bottleneck router be t_p. If the bottleneck link has been busy for at least r seconds, and a packet just arrived at the congested router at time t, then the queue length at the congested router is*

$$Q(t) = w(t - t_p) - rd \tag{7.1}$$

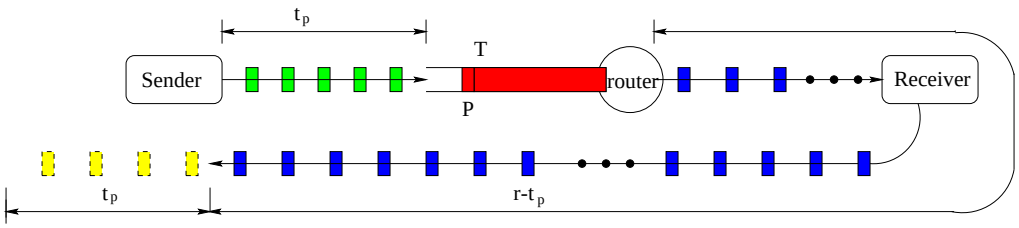


Figure 7.1: Calculation of the queue length

Proof Consider the packet that just arrived at the congested router at time t. It was sent by the source at time $t - t_p$. At that time, the number of packets on the path and outstanding ACKs on the reverse link was $w(t - t_p)$. By time t, $t_p d$ ACKs are received by the source. All packets between the source and the router have entered the congested router or have been sent downstream. As shown in Figure 7.1, the pipe length from the congested router to the receiver, and then back to the source is $r - t_p$. The number of downstream packets and outstanding ACKs are $(r - t_p)d$. The rest of the $w(t - t_p)$ unacknowledged packets are still in the congested router. So the queue length is

$$Q(t) = w(t - t_p) - t_p d - (r - t_p)d = w(t - t_p) - rd \tag{7.2}$$

This completes the proof.

Notice that in this theorem, we did not use the number of packets between the source and the congested router to estimate the queue length, because the packets downstream from the congested router and the ACKs on the reverse link are equally spaced, but the same may not be true for packets between the source and the congested router.

The analysis in this theorem is based on the assumptions in section 2. The conclusion applies to both slow start and congestion avoidance phases. In order for equation (7.1) to hold, the router must have been congested for at least r seconds.

7.4 Buffer Size Requirement and Threshold Setting

When ECN signals are used for congestion control, the network can achieve zero packet loss. When acknowledgments are not marked, the source gradually increase the window size. Upon the receipt of an ECN-Echo, the source halves its congestion window to reduce the congestion.

In this section, we analyze the buffer size requirement for both mark-tail and mark-front strategies. The result also includes an analysis on how to set the threshold.

7.4.1 Mark-Tail Strategy

Suppose P was the packet that increased the queue length over the threshold T, and it was sent from the source at time s_0 and arrived at the congested router at time t_0. Its acknowledgment, which was an ECN-Echo, arrived at the source at time s_1 and the window

was reduced at the same time. We also assume that the last packet before the window reduction was sent at time s_1^- and arrived at the congested router at time t_1^-.

In order to use Theorem 1, we need to consider two cases separately: when T is large and when T is small, as compared to rd.

Case 1 If T is reasonably large (about rd) such that the buildup of a queue of size T needs r time, the assumption in Theorem 1 is satisfied, we have

$$T = Q(t_0) = w(t_0 - t_p) - rd = w(s_0) - rd \tag{7.3}$$

so

$$w(s_0) = T + rd \tag{7.4}$$

Since the time elapsed between s_0 and s_1 is one RTT, if packet P were not marked, the congestion window would increase to $2w(s_0)$. Since P was marked, the congestion window before receiving the ECN-Echo was

$$w(s_1^-) = 2w(s_0) - 1 = 2(T + rd) - 1 \tag{7.5}$$

When the last packet sent under this window reached the router at time t_1^-, the queue length was

$$Q(t_1^-) = w(s_1^-) - rd = 2w(s_0) - 1 - rd = 2T + rd - 1 \tag{7.6}$$

Upon the receipt of ECN-Echo, the congestion window was halved. The source can not send any more packets before half of the packets are acknowledged. So $2T + rd - 1$ is the maximum queue length.

Case 2 If T is small, rd is an overestimate of the number of downstream packets and ACKs on the reverse link.

$$w(s_0) = T + \textit{number of downstream packets and ACKs } \leq T + rd \tag{7.7}$$

Therefore,

$$Q(t_1^-) = w(s_1^-) - rd = (2w(s_0) - 1) - rd \leq 2(T + rd) - 1 - rd = 2T + rd - 1 \tag{7.8}$$

So, in both cases, $2T + rd - 1$ is an upper bound of queue length that can be reached in slow start phase.

Theorem 2 *In a TCP connection with ECN congestion control, if the fixed round trip time is r seconds, the bottleneck link rate is d packets per second, and the bottleneck router uses threshold T for congestion detection, then the maximum queue length that can be reached in slow start phase is less than or equal to $2T + rd - 1$.*

As shown by equation (7.6), when T is large, the bound $2T + rd - 1$ can be reached with equality. When T is small, $2T + rd - 1$ is just an upper bound. Since the queue length in congestion avoidance phase is smaller, this bound is actually the buffer size requirement.

7.4.2 Mark-Front Strategy

Suppose P was the packet that increased the queue length over the threshold T, and it was sent from the source at time s_0 and arrived at the congested router at time t_0. The router marked the packet P' that stood in the front of the queue. The acknowledgment of P', which was an ECN-Echo, arrived at the source at time s_1 and the window was reduced at the same time. We also suppose the last packet before the window reduction was sent at time s_1^- and arrived at the congested router at time t_1^-.

Consider two cases separately: when T is large and when T is small.

Case 1 If T is reasonably large (about rd) such that the buildup of a queue of size T needs r time, the assumption in Theorem 1 is satisfied. We have

$$T = Q(t_0) = w(t_0 - t_p) - rd = w(s_0) - rd \tag{7.9}$$

so

$$w(s_0) = T + rd \tag{7.10}$$

In slow start phase, the source increases the congestion window by one for every acknowledgment it receives. If the acknowledgment of P was received at the source without the congestion indication, the congestion window would be doubled to

$$2w(s_0) = 2(T + rd).$$

However, when the acknowledgment of P' arrived, $T - 1$ acknowledgments corresponding to packets prior to P were still on the way. So the window size at time s_1^- was

$$w(s_1^-) = 2w(s_0) - (T - 1) - 1 = T + 2rd \tag{7.11}$$

When the last packet sent under this window reached the router at time t_1^-, the queue length was

$$Q(t_1^-) = w(s_1^-) - rd = T + 2rd - rd = T + rd \tag{7.12}$$

Upon the receipt of ECN-Echo, congestion window is halved. The source can not send any more packets before half of the packets are acknowledged. So $T + rd$ is the maximum queue length.

Case 2 If T is small, rd is an overestimate of the number of downstream packets and ACKs on the reverse link.

$$w(s_0) = T + \textit{number of downstream packets and ACKs } \leq T + rd \tag{7.13}$$

Therefore,

$$Q(t_1^-) = w(s_1^-) - rd = (2w(s_0) - T) - rd \leq 2(T + rd) - T - rd = T + rd \tag{7.14}$$

So, in both cases, $T + rd$ is an upper bound of queue length that can be reached in the slow start phase.

Theorem 3 *In a TCP connection with ECN congestion control, if the fixed round trip time is r seconds, the bottleneck link rate is d packets per second, and the bottleneck router uses threshold T for congestion detection, then the maximum queue length that can be reached in slow start phase is less than or equal to $T + rd$.*

Again, when T is large, equation (7.12) shows the bound $T + rd$ is tight. Since the queue length in congestion avoidance phase is smaller, this bound is actually the buffer size requirement.

Theorem 2 and 3 estimate the buffer size requirement for zero-loss ECN congestion control.

7.4.3 Threshold Setting

In the congestion avoidance phase, the congestion window increases roughly by one in every RTT. Assuming the mark-tail strategy is used, using the same timing variables as in the previous subsections, we have

$$w(s_0) = Q(t_0) + rd = T + rd \tag{7.15}$$

The congestion window increases roughly by one in an RTT,

$$w(s_1^-) = T + rd + 1 \tag{7.16}$$

When the last packet sent before the window reduction arrived at the router, it saw a queue length of $T + 1$:

$$Q(t_1^-) = w(s_1^-) - rd = T + 1 \tag{7.17}$$

Upon the receipt of the ECN-Echo, the window was halved:

$$w(s_1) = (T + rd + 1)/2 \tag{7.18}$$

The source may not be able to send packets immediately after s_1. After some packets were acknowledged, the halved window allowed new packets to be sent. The first packet sent under the new window saw a queue length of

$$Q(t_1) = w(s_1) - rd = (T + rd + 1)/2 - rd = (T - rd + 1)/2 \tag{7.19}$$

The congestion window was fixed for an RTT and then began to increase. So $Q(t_1)$ was the minimum queue length in a cycle.

In summary, in the congestion avoidance phase, the maximum queue length is $T + 1$ and the minimum queue length is $(T - rd + 1)/2$.

In order to avoid link idling, we should have $(T - rd + 1)/2 \geq 0$ or equivalently, $T \geq rd - 1$. On the other hand, if $minQ$ is always positive, the router keeps an unnecessarily large queue and all packets suffer a long queueing delay. Therefore, the best choice of threshold should satisfy

$$(T - rd + 1)/2 = 0 \tag{7.20}$$

or

$$T = rd - 1 \tag{7.21}$$

If the mark-front strategy is used, the source's congestion window increases roughly by one in every RTT, but congestion feedback travels faster than the data packets. Hence

$$Q(s_1^-) = T + rd + \epsilon \tag{7.22}$$

where ϵ is between 0 and 1, and depends on the location of the congested router. Therefore,

$$Q(t_1^-) = w(s_1^-) - rd = T + \epsilon \tag{7.23}$$

$$w(s_1) = (T + rd + \epsilon)/2 \tag{7.24}$$

$$Q(t_1) = w(s_1) - rd = (T + rd + \epsilon)/2 - rd = (T - rd + \epsilon)/2 \tag{7.25}$$

For the reason stated above, the best choice of threshold is $T = rd - \epsilon$. Compared with rd, the difference between $rd - \epsilon$ and $rd - 1$ can be ignored. So we have the following theorem:

Theorem 4 *In a path with only one connection, the optimal threshold that achieves full link utilization while keeping the queueing delay minimal in congestion avoidance phase is* $rd - 1$. *If the threshold is smaller than this value, the link will be under-utilized. If the threshold is greater than this value, the link can be fully utilized, but packets will suffer an unnecessarily large queueing delay.*

Combining the results in Theorem 2, 3 and 4, we can see that the mark-front strategy reduces the buffer size requirement from about $3rd$ to $2rd$. It also reduces the congestion feedback's delay by one fixed round-trip time.

7.5 Lock-out Phenomenon and Fairness

One of the weaknesses of mark-tail policy is its discrimination against new flows. Consider the time when a new flow joins the network, but the buffer of the congested router

is occupied by packets of old flows. In the mark-tail strategy, the packet that just arrived will be marked, but the packets already in the buffer will be sent without being marked. The acknowledgments of the sent packets will increase the window size of the old flows. Therefore, the old flows which already have a large share of the resources will grow even larger. However, the new flow with small or no share of the resources has to back off, since its window size will be reduced by the marked packets. This causes a "lock-out" phenomenon in which a single connection or a few flows monopolize the buffer space and prevent other connections from getting room in the queue [24]. Lock-out leads to gross unfairness among connections and is clearly undesirable.

Contrary to the mark-tail policy, the mark-front strategy marks the packets in the buffer first. Connections with large buffer occupancy will have more packets marked than connections with small buffer occupancy. Compared with the mark-tail strategy that let the packets in the buffer escape the marking, the mark-front strategy helps to prevent the lock-out phenomenon. Therefore, we can expect that the mark-front strategy to be fairer than mark-tail strategy.

TCP's discrimination against connections with large RTT is also well known. The cause of this discrimination is similar to the discrimination against new connections. If connections with small RTT and large RTT start at the same time, the connections with small RTT will receive their acknowledgment faster and therefore grow faster. When congestion happens, connections with small RTT will take more buffer room than connections with large RTT. With mark-tail policy, packets already in the queue will not be marked, only newly arrived packets will be marked. Therefore, connections with small RTT will grow even

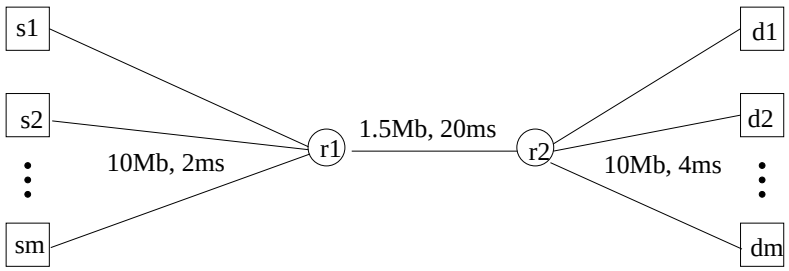


Figure 7.2: Simulation model

larger, but connections with large RTT will have to back off. Mark-front alleviates this discrimination by treating all packets in the buffer equally. Packets already in the buffer may also be marked. Therefore, connections with large RTT can have larger bandwidth shares.

7.6 Simulation Results

In order to compare the mark-front and mark-tail strategies, we performed a set of simulations with the ns simulator [53]. We modified the RED algorithm in ns simulator to deterministically mark the packets when the real queue length exceeds the threshold. The basic simulation model is shown in Figure 7.2. A number of sources $s_1, s_2, \ldots, s_m$ are connected to the router r_1 by 10 Mbps links, router r_1 is connected to r_2 by a 1.5 Mbps link, and destinations $d_1, d_2, \ldots, d_m$ are connected to r_2 by 10 Mbps links. The link speeds are chosen so that congestion will only happen at the router r_1, where mark-tail and mark-front strategies are tested.

With the basic configuration shown in Figure 7.2, the fixed round trip time, including the propagation time and the transmission time at the routers, is 59 ms. Changing the propagation delay between router r_1 and r_2 from 20 ms to 40 ms gives an RTT of 99 ms. Changing the propagation delays between the sources and router r_1 gives us configurations

of different RTT. An FTP application runs on each source. Reno TCP and ECN are used for congestion control. The data packet size, including all headers, is 1000 bytes and the acknowledgment packet size is 40 bytes.

With the basic configuration,

$$rd = 0.059 \times 1.5 \times 10^6 \text{ bits } = 11062.5 \text{ bytes } \approx 11 \text{ packets}$$

In our simulations, the routers are configured to implement mark-tail or mark-front. The results for both strategies are compared.

7.6.1 Simulation Scenarios

In order to show the difference between mark-front and mark-tail strategies, we designed the following simulation scenarios based on the basic simulation model described in Figure 7.2. If not specified, all connections have an RTT of 59 ms, start at the zeroth second and stop at the 10th second.

1. One connection.
2. Two connections with the same RTT.
3. Two overlapping connections with the same RTT, but the first connection starts at the zeroth second and stops at the ninth second, the second connection starts at the first second and stops at the 10th second.
4. Two connections with RTT equal to 59 and 157 ms respectively.
5. Two connections with same RTT, but the buffer size at the congested router is limited to 25 packets.
6. Five connections with the same RTT.

7. Five connections with RRT of 59, 67, 137, 157 and 257 ms respectively.

8. Five connections with the same RTT, but the buffer size at the congested router is limited to 25 packets.

Scenarios 1, 4, 6 and 7 are mainly designed for testing the buffer size requirement. Scenarios 1, 3, 4, 6, 7, 8 are for link efficiency, and scenarios 2, 3, 4, 5, 6, 7 are for fairness among connections.

7.6.2 Metrics

We use three metrics to compare the two strategies. The first metric is the *buffer size requirement* for zero loss congestion control. This is the maximum queue size that can be built up at the router in the slow start phase before the congestion signal takes effect at the congested router. If the buffer size is greater or equal to this value, no packet loss will happen. This metric is measured as the maximum queue length in the entire simulation.

The second metric, *link efficiency*, is calculated from the number of acknowledged packets (not counting the retransmissions) divided by the possible number of packets that can be transmitted during the simulation. Because of the slow start phase and possible link idling after the window reduction, the link efficiency is always smaller than 1. Link efficiency should be measured with long simulation duration to minimize the effect of the initial transient state. We tried different durations from 5 seconds to 100 seconds. The results for 10 seconds show the essential features of the strategy, without much difference from the results for 100 seconds. The simulation results presented in this chapter are based on 10-second simulations.

The third metric, *fairness* index, is calculated according to the formula in [42]. If m connections share the bandwidth, and x_i is the number of acknowledged packets of

connection i, then the *fairness* index is calculated as:

$$fairness = \frac{(\sum_{i=1}^{m} x_i)^2}{m \sum_{i=1}^{m} x_i^2} \tag{7.26}$$

Because the fairness index is often close to 1, in our graphs, we draw the $unfairness$ index:

$$unfairness = 1 - fairness \tag{7.27}$$

The performance of ECN depends on the selection of the threshold value. In our results, all three metrics are drawn for different values of threshold.

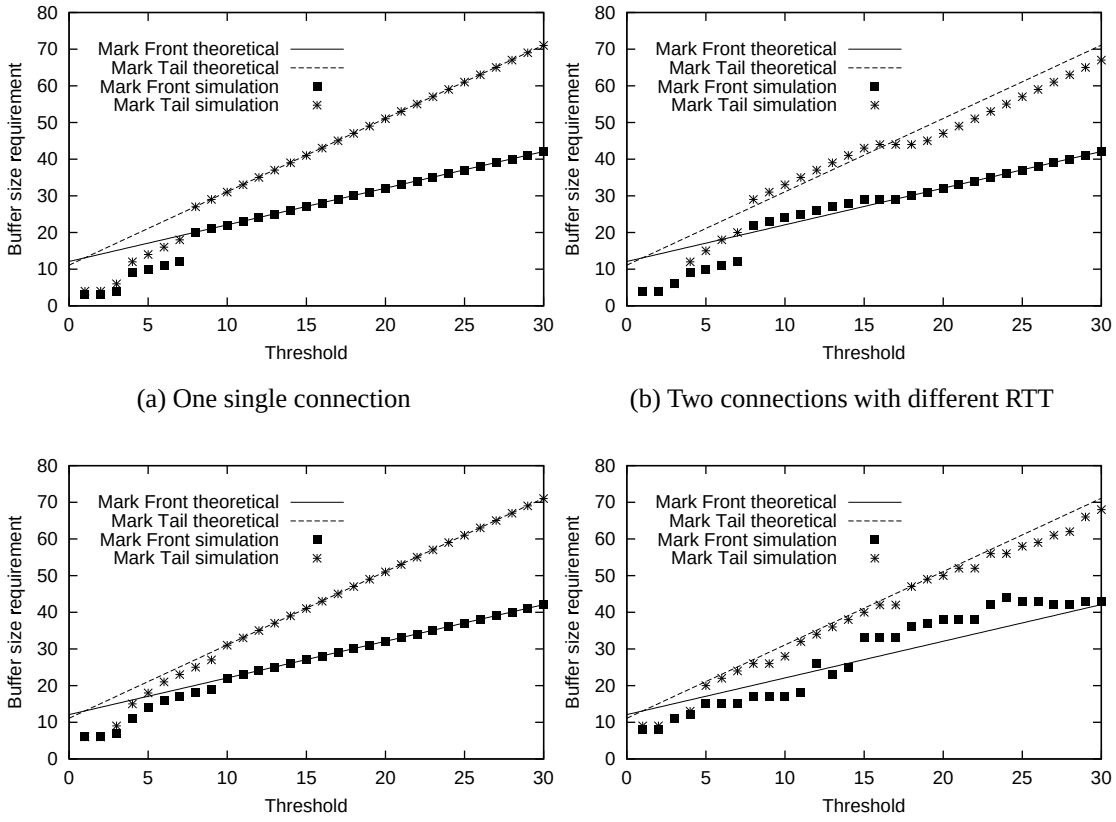
7.6.3 Buffer Size Requirement

Figure 7.3 shows the buffer size requirement for mark-tail and mark-front. The measured maximum queue lengths are shown with "□" and "△" respectively. The corresponding theoretical estimates from Theorem 2 and 3 are shown with dashed and solid lines. In Figure 7.3(b) and (d), where the connections have different RTT, the theoretical estimate is calculated from the smallest RTT.

From the simulation, we find that for connections with the same RTT, the theoretical estimate of buffer size requirement is accurate. When threshold T is small, the buffer size requirement is an upper bound, when $T \geq rd$, the upper bound is tight. For connections with different RTT, the estimate given by the largest RTT is an upper bound, but is usually an over estimate. The estimate given by the smallest RTT is a closer approximation.

7.6.4 Link Efficiency

Figure 7.4 shows the link efficiency for various scenarios. In all cases, the efficiency increases with the threshold, until the threshold is about rd, where the link reaches almost



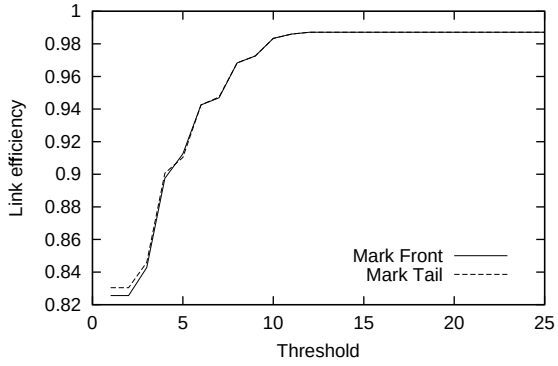
(a) One single connection

(b) Two connections with different RTT

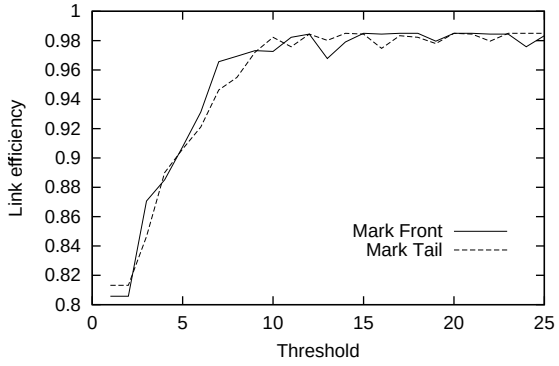
(c) Five connections with same RTT

(d) Five connections with different RTT

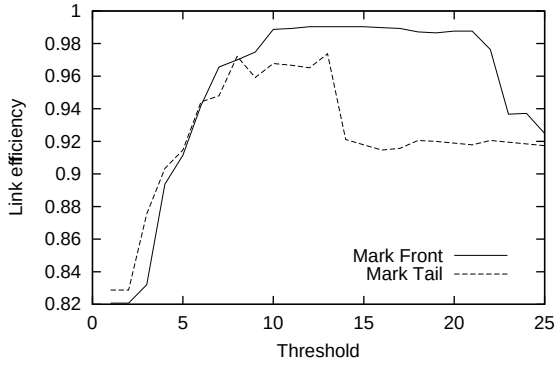
Figure 7.3: Buffer size requirement in various scenarios



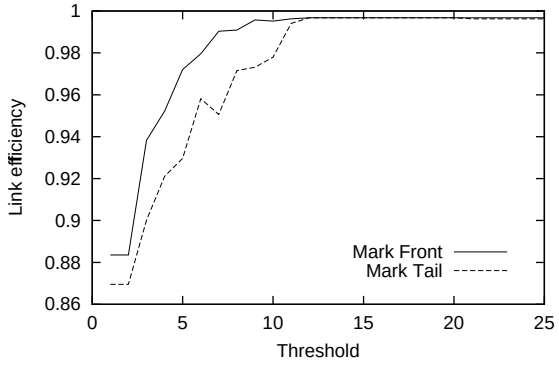
(a) One single connection



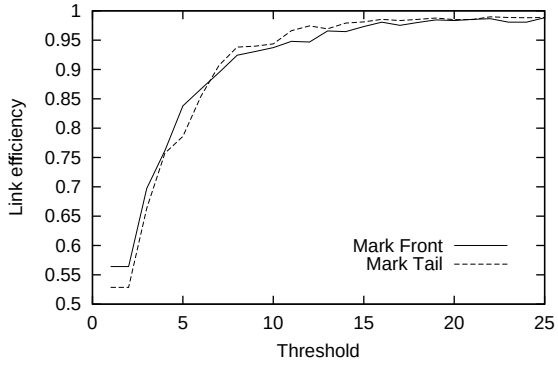
(b) Two overlapping connections



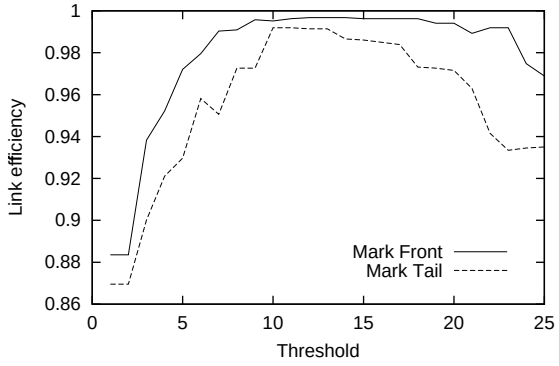
(c) Two same connections, limited buffer



(d) Five connections with same RTT

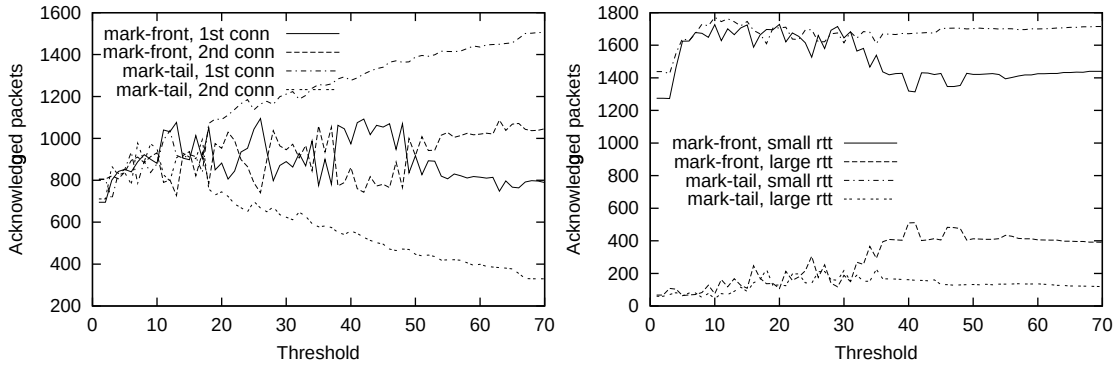


(e) Five connections with different RTT



(f) Five same connections, limited buffer

Figure 7.4: Link efficiency in various scenarios



(a) Two overlapping connections

(b) Two connections with different RTT

Figure 7.5: Lock-out phenomenon and alleviation by the mark-front strategy

full utilization. A small threshold results in low link utilization because it generates congestion signals even when the router is not really congested. Unnecessary window reduction actions taken by the source lead to link idling. The link efficiency results in Figure 7.4 verify the choice of threshold stated in Theorem 4.

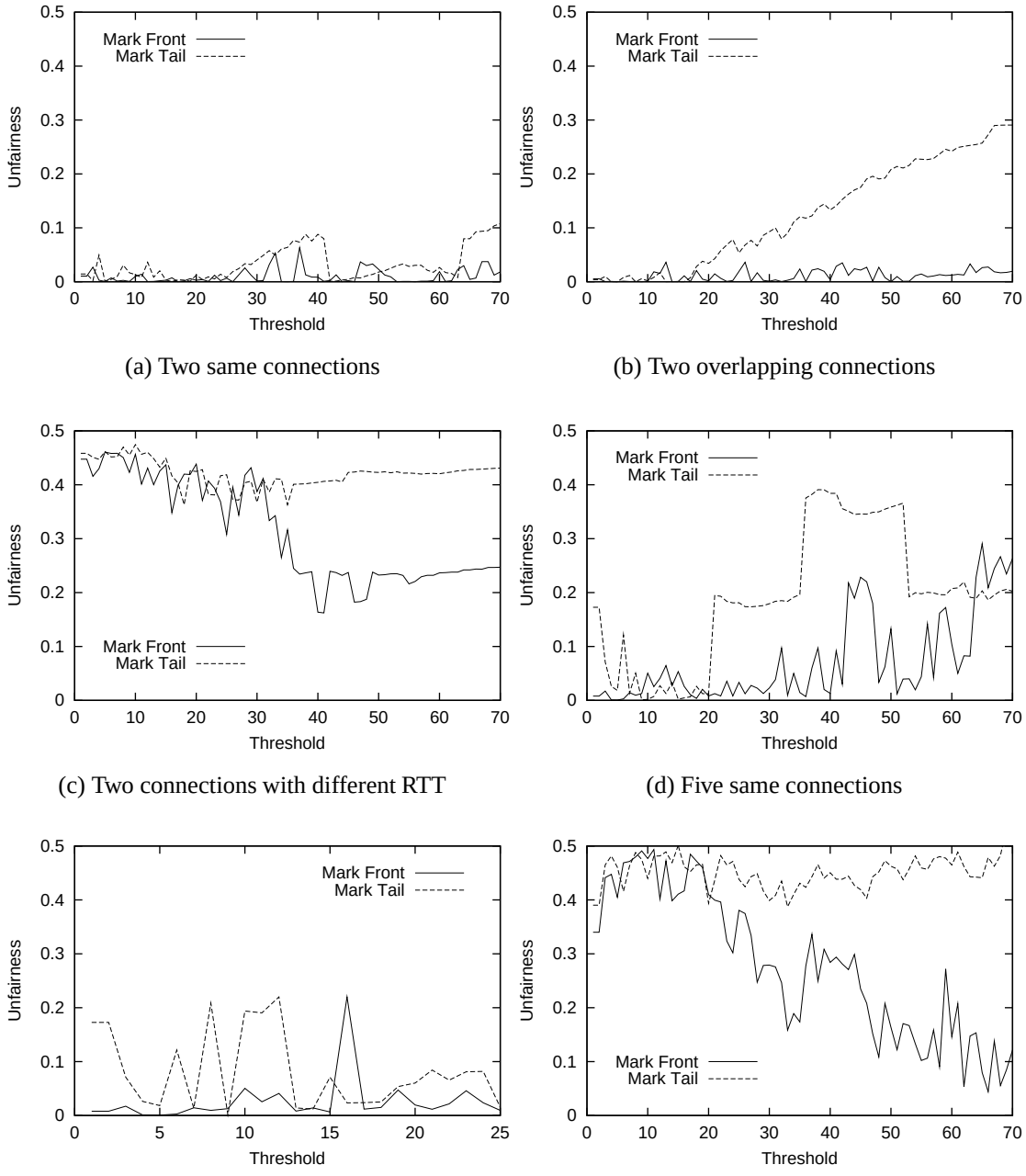
In the unlimited buffer cases (a), (b), (d), (e), the difference between mark-tail and mark-front is small. However, when the buffer size is limited as in cases (c) and (f), mark-front has much better link efficiency. This is because when congestion occurs, the mark-front strategy provides a faster congestion feedback than mark-tail. Faster congestion feedback prevents the source from sending more packets that will be dropped at the congested router. Multiple drops cause source timeout and idling at the bottleneck link, and thus the low utilization. This explains the drop of link efficiency in Figure 7.4 (c) and (f) when the threshold exceeds about 10 packets for mark-tail and about 20 packets in mark-front.

7.6.5 Fairness

Scenarios 2 – 7 are designed to test the fairness of the two marking strategies. Figure 7.5 shows lock-out phenomenon and alleviation by the mark-front strategy. With the mark-tail strategy, old connections occupy the buffer and lock-out new connections. Although the two connections in scenario 3 have the same time span, the number of the acknowledged packets in the first connection is much larger than that of the second connection, Figure 7.5(a). In scenario 4, the connection with large RTT (157 ms) starts at the same time as the connection with small RTT (59 ms), but the connection with small RTT grows faster, takes over a large portion of the buffer room, and locks out the connection with large RTT. Of all of the bandwidth, only 6.49% is allocated to the connection with large RTT. The mark-front strategy alleviates the discrimination against large RTT by marking packets already in the buffer. Simulation results show that the mark-front strategy improves the portion of bandwidth allocated to connection with large RTT from 6.49% to 21.35%.

Figure 7.6 shows the unfairness index for the mark-tail and the mark-front strategies. In Figure 7.6(a), the two connections have the same configuration. Which connection receives more packets than the other is not deterministic, so the unfairness index seems random. But in general, mark-front has smaller unfairness index than mark-tail.

In Figure 7.6(b), the two connections are different: the first connection starts sooner and takes the buffer room. Although the two connections have the same time span, if the mark-tail strategy is used, the second connection is locked out by the first and therefore receives fewer packets. Mark-front avoids this lock-out phenomenon. The results show that the unfairness index of mark-front is much smaller than that of mark-tail. In addition, as the threshold increases, the unfairness index of mark-tail increases, but the mark-front remains roughly the same, regardless of the threshold.



(a) Two same connections

(b) Two overlapping connections

(c) Two connections with different RTT

(d) Five same connections

(e) Five same connections, limited buffer

(f) Five connections with different RTT

Figure 7.6: Unfairness in various scenarios

Figure 7.6(c) shows the difference between connections with different RTT. With the mark-tail strategy, the connections with small RTT grow faster and therefore locked out the connections with large RTT. Since the mark-front strategy does not have the lock-out problem, the discrimination against connections with large RTT is alleviated. The difference of the two strategies is obvious when the threshold is large.

Figure 7.6(e) shows the unfairness index when the router buffer size is limited. In this scenario, when the buffer is full, the router drops the the packet in the front of the queue. Whenever a packet is sent, the router checks whether the current queue size is larger than the threshold. If yes, the packet is marked. The figure shows that mark-front is fairer than mark-tail.

Similar results for five connections are shown in Figure 7.6(d) and 7.6(f).

7.7 Apply to RED

The analytical and simulation results obtained in previous sections are based on the simplified congestion detection model which marks a packet leaving a router if the actual queue size of the router exceeds the threshold. However, RED uses a different congestion detection criterion. First, RED uses average queue size instead of the actual queue size. Second, a packet is not marked deterministically, but with a probability calculated from the average queue size.

In this section, we apply the mark-front strategy to the RED algorithm and compare the results with the mark-tail strategy. Because of the difficulty in analyzing RED mathematically, the comparison is carried out by simulations only.

The RED algorithm needs four parameters: queue weight w, minimum threshold th_{min}, maximum threshold th_{max} and maximum marking probability p_{max}. Although determining

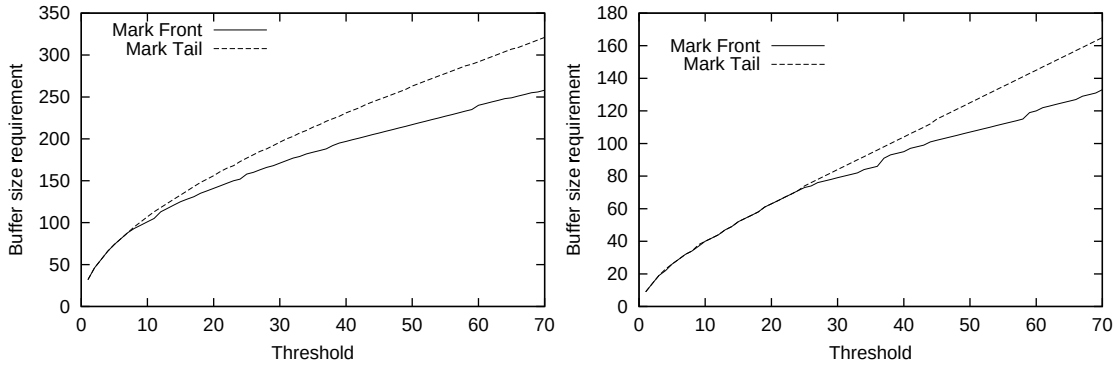
the best RED parameters is out of the scope of this chapter, we have tested several hundred of combinations. In almost all these combinations, mark-front has better performance than mark-tail in terms of buffer size requirement, link efficiency and fairness.

Instead of presenting individual parameter combinations for all scenarios, we focus on one scenario and present the results for a range of parameter values. The simulation scenario is two overlapping connections described in section 6.1, scenario 3. Based on the recommendations in [31], we vary the queue weight w for four values: 0.002, 0.02, 0.2 and 1, vary th_{min} from 1 to 70, fix th_{max} as $2th_{min}$, and fix p_{max} as 0.1.

Figure 7.7 shows the buffer size requirement for both strategies with different queue weight. In all cases, the mark-front strategy requires a smaller buffer size than the mark-tail. The results also show that queue weight w is a major factor affecting the buffer size requirement. Smaller queue weights require larger buffers. When the actual queue size is used (corresponding to $w = 1$), RED requires the minimum buffer size.

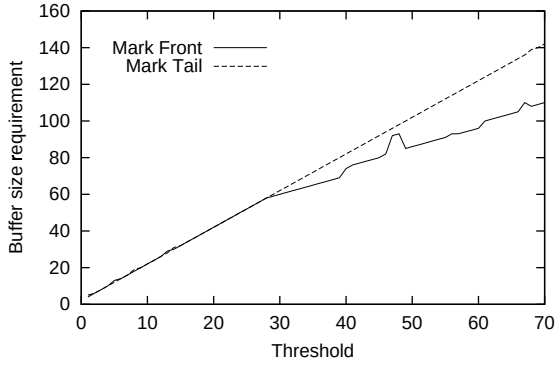
Figure 7.8 shows the link efficiency. For almost all values of threshold, mark-front provides better link efficiency than mark-tail. Contrary to the common belief, the actual queue size (Figure 7.8(d)) is no worse than the average queue size (Figure 7.8(a)) in achieving higher link efficiency.

The queue size trace at the congested router shown in Figure 7.9 provides some explanation for the smaller buffer size requirement and higher efficiency of the mark-front strategy. The RED parameters for this simulation are $w = 0.002, th_{min} = 70, th_{max} = 140, p_{max} = 0.1$. When congestion happens, mark-front delivers faster congestion feedback than mark-tail so that the sources can stop sending packets earlier. In Figure 7.9(a), with the mark-tail signal, the queue size stops increasing at 1.98 second. With mark-front

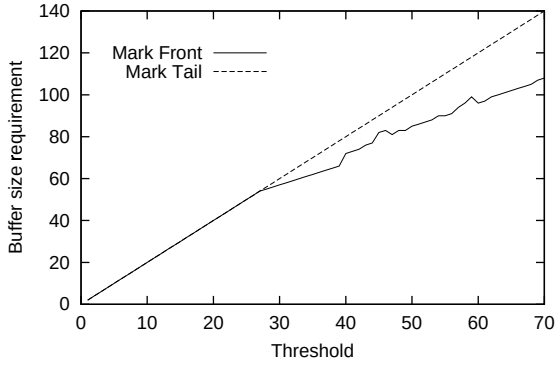


(a) w=0.002

(b) w=0.02

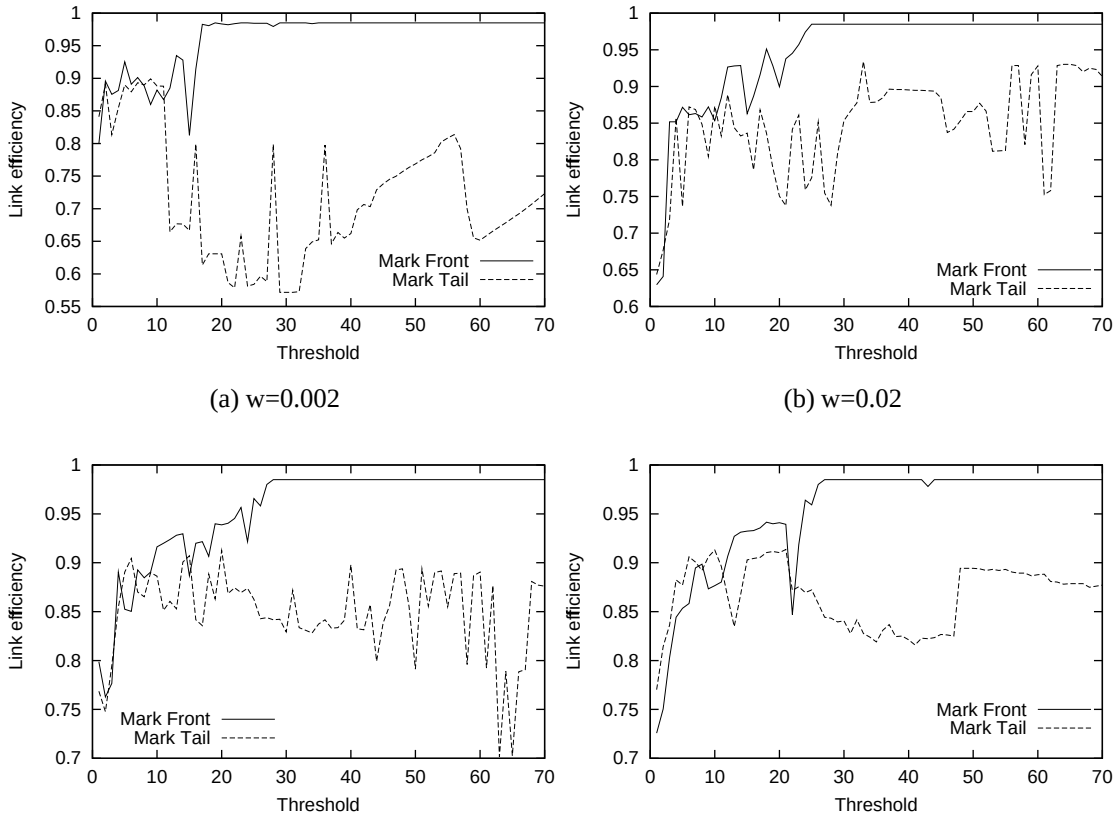


(c) w=0.2



(d) w=1

Figure 7.7: Buffer size requirement for different queue weight, $p_{max} = 0.1$



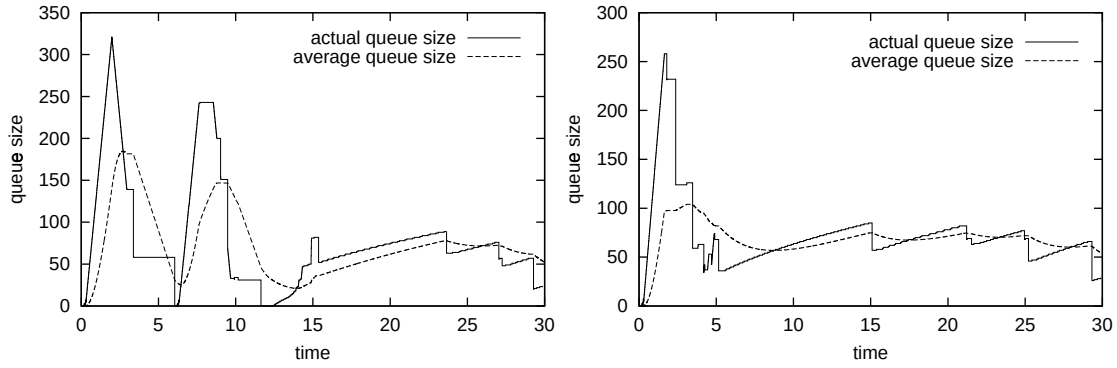
(a) w=0.002

(b) w=0.02

(c) w=0.2

(d) w=1

Figure 7.8: Link efficiency for different queue weight, $p_{max} = 0.1$



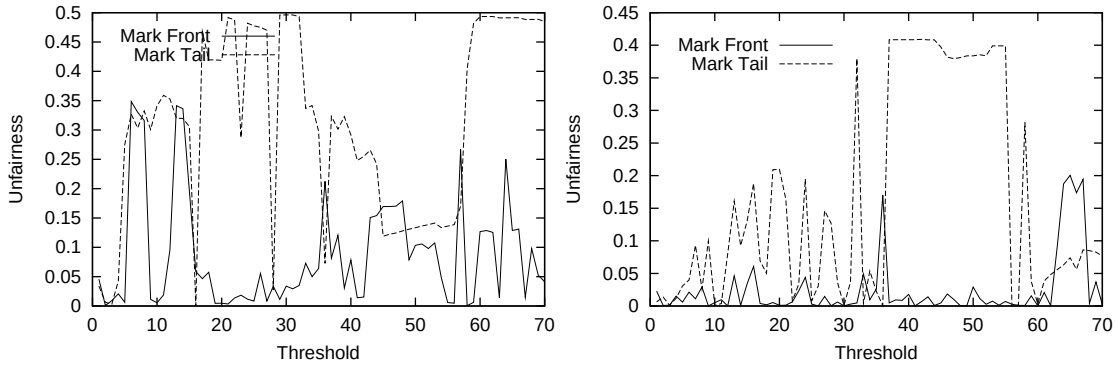
(a) Mark-tail (b) Mark-front

Figure 7.9: Change of queue size at the congested router

signal, the queue size stops increasing at 1.64 second. Therefore, the mark-front strategy needs a smaller buffer.

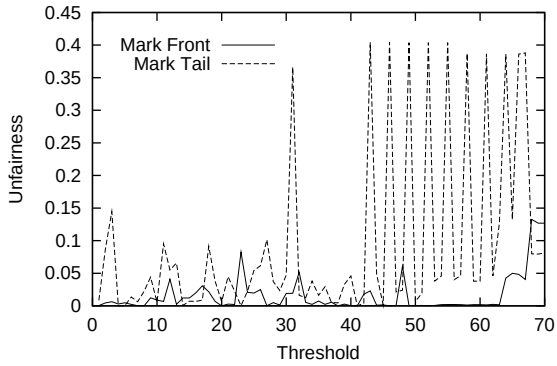
On the other hand, when congestion is gone, mark-tail is slow in reporting the change of congestion status. Packets leaving the router still carry the congestion information set at the time when they entered the queue. Even if the queue is empty, these packets still tell the sources that the router is congested. This out-dated congestion information is responsible for the link idling around the 6th second and the 12th second in Figure 7.9(a). As a comparison, in Figure 7.9(b), the same packets carry more up-to-date congestion information to tell the sources that the router is no longer congested, so the sources send more packets. Thus the mark-front signal helps to avoid link idling and improve the efficiency.

Figure 7.10 shows the unfairness index. Both mark-front and mark-tail have big oscillations in the unfairness index when the threshold changes. These oscillations are caused by the randomness of the number of packets of each connection getting marked in the bursty TCP slow start phase. Changing the threshold value can significantly change the number of

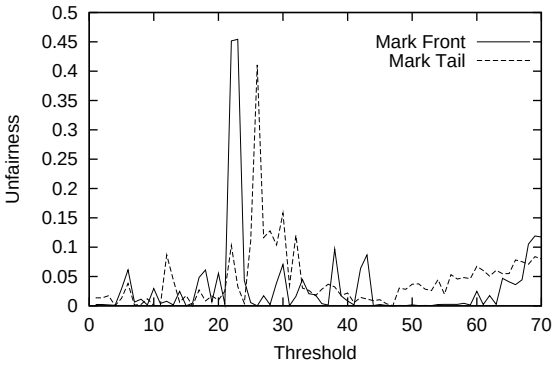


(a) w=0.002

(b) w=0.02



(c) w=0.2



(d) w=1

Figure 7.10: Unfairness for different queue weight, $p_{max} = 0.1$

marked packets of each connection. In spite of the randomness, in most cases mark-front is fairer than mark-tail.

7.8 Chapter Summary

In this chapter we analyze the mark-front strategy used in ECN. Instead of marking the packet from the tail of the queue, this strategy marks the packet in the front of the queue and thus delivers faster congestion signals to the source. Compared with the mark-tail policy, mark-front strategy has three advantages. First, it reduces the buffer size requirement at

the routers. Second, it provides more up-to-date congestion information to help the source adjust its window in time to avoid packet losses and link idling, and thus improves the link efficiency. Third, it improves the fairness among old and new connections, and helps to alleviate TCP's discrimination against connections with large round trip time.

With a simplified model, we analyze the buffer size requirement for both mark-front and mark-tail strategies. Link efficiency, fairness and more complicated scenarios are tested with simulations. The results show that the mark-front strategy achieves better performance than the current mark-tail policy. We also apply the mark-front strategy to the RED algorithm. Simulations show that mark-front strategy used with RED has similar advantages over mark-tail.

Based on the analysis and the simulations, we conclude that mark-front is an easy-to-implement improvement that provides a better congestion control and helps TCP to achieve smaller buffer size requirements, higher link efficiency and better fairness among connections.

CHAPTER 8

IMPROVING RESOURCE UTILIZATION IN ASYMMETRIC WIRELESS LANS

In this chapter, we document our design of a MAC protocol, called *OSU-MAC*, subject to the physical layer characteristics and constraints of a narrow-band wireless modem testbed currently being built at the Ohio State University. The narrow-band wireless modem testbed is expected to support both real-time (bus location tracking) and non-real-time (regular) data applications. A number of techniques are proposed to support QoS imposed by the real-time applications, to deal with the asymmetry on the forward and reverse channels and the half-duplex transmission constraint imposed by the physical layer, and to enhance the error control capability of *OSU-MAC*. We also present simulation results to demonstrate the key, functional characteristics of *OSU-MAC*.

8.1 Introduction

Wireless communication has become an important technique for supporting emerging Personal Communications Service (PCS). In PCS, both traditional telephone service and other more advanced data applications, e.g., audio/video, and periodic update of sensor information, are expected to be simultaneously supported. The latter applications, in particular, require different levels of temporal quality of service (QoS). An effective medium

access control (MAC) protocol must be carefully designed for this purpose. In this chapter, we document the design of a MAC protocol, called *OSU-MAC*, that supports both real-time and non-real-time applications on an OSU narrow-band wireless modem testbed, subject to its physical layer characteristics and constraints. The narrow-band wireless modem testbed is expected to support both real-time and non-real-time (regular) data applications, currently with the real-time bus location tracking via an on-board global positioning system (GPS) being the representative, real-time application and data applications such as e-mail, ftp, and telnet, being the other representatives.

Our first step toward realizing the above objective is to design and implement an effective MAC protocol, *OSU-MAC*, that coordinates the transmission activities on the forward and reverse channels so as to support QoS requirements imposed by both types of applications. In particular, we delegate to the base station the full responsibility of resource arbitration, channel access, and registration in each cell. As will be elaborated on in Section 8.3.1, this base- station-based scheduling approach enables provisioning of deterministic QoS for real-time applications, while maximizing the system utilization. To take the error characteristics of the physical layer into consideration, we encode both data packets and control fields in Reed-Solomon code [10, 58]. We also take into account the half-duplex transmission constraint (i.e., a mobile subscriber cannot transmit and receive at the same time) imposed by the physical layer, and propose a two-control-field structure to fully utilize the limited bandwidth available on the reverse channel. To make use of the unused bandwidth originally reserved for real-time traffic, we also propose a dynamic slot adjustment scheme. Finally, as a real-life MAC protocol currently being implemented on

the OSU narrow-band wireless modem testbed, we insert, wherever advised by the wireless modem researcher, preambles, postambles, and guard times between packet slots or notification cycles for synchronization.

Several MAC protocols have been proposed in wireless networks, among which Packet Reservation Multiple Access (PRMA) [59], Dynamic TDMA (D-TDMA) [74], Dynamic Reservation Multiple Access (DRMA) [64], Resource Auction Multiple Access (RAMA) [3], Floor Acquisition Multiple Access (FAMA) [19], Remote-Queuing Multiple Access (RQMA) [25], and Multimedia Cable Network System (MCNS) [15] may have received the most attention. They have been designed either for voice/data traffic (PRMA, D-TDMA, DRMA, RAMA, and FAMA), for real-time/best-effort traffic on an abstract model (RQMA), or for cable modem users (MCNS). In comparison with the above research efforts, OSU-MAC is the first implementation work that (i) takes into account the physical layer characteristics and constraints on a specific environment in the design phase (rather than an abstract design), (ii) provides mechanisms for providing deterministic, temporal QoS for real-time applications, and (iii) is fault tolerant with both data slots and control fields protected by the (64,48) Reed-Solomon code.

The rest of the chapter is organized as follows. In Section 8.2, we introduce the design objectives of, the types of applications to be supported on, and the system model used for, the OSU narrow-band wireless testbed. We also outline the physical layer characteristics and constraints of the wireless testbed. These characteristics and constraints are the major factors in directing the design of the proposed MAC protocol. In Section 8.3, we present *OSU-MAC*. In Section 2.6, we give a survey of existing MAC protocols for wireless local area networks. In Section 8.4, we evaluate *OSU-MAC* in terms of throughput, packet delay,

capability of providing temporal QoS, collision probability, and registration latency. We conclude the chapter with Section 8.5.

8.2 The OSU Narrow-Band Wireless Testbed

8.2.1 Applications supported and design goals

Applications supported: Two types of applications will be supported: one is real-time bus location tracking via an on-board global positioning system (GPS). In the GPS system, short GPS packets of size no more than 72 bits are periodically sent from each bus being tracked to report the location of the unit. Timely delivery of these packets is important in correctly tracking the bus location. The other applications to be supported are those supported by TCP/IP, e.g., e-mail message retrieval, FTP and Telnet.

Design objectives: In the current narrow-band wireless modem testbed, each base station covers an area about 10 km in radius and is expected to support

- Up to 8 active GPS users with 1 minute checking delay and 4 second access delay requirements, where the checking delay is the delay incurred when a non-active terminal becomes active, and the access delay is the time interval between the arrival of a packet at an active GPS user and its transmission.

  The 4 second access delay requirement is determined as follows: since a mobile unit moves at a speed no faster than 90 kilometers per hour and the allowable location error is within 100 meters, a GPS packet must be sent and received every $100/(90 \times 10^3)$ hours, or 4 seconds. If a GPS packet is corrupted after being decoded, the corrupted packet will not be retransmitted.

- Up to 64 active non-real-time users. No access delay requirement is imposed, but data packets need to be reliably transported.

By active users, we mean mobile subscribers which have registered with the base station and are actively transmitting/receiving data packets. The maximum time required to register with the base station is also a design parameter. In the current design, we require that 80% of the registration requests can be approved in two notification cycles, and 99% can be made in 10 cycles.

8.2.2 System model

The geographical area covered by a wireless network is divided into overlapping cells. Each cell is associated with a base station and can support a number of mobile subscribers. The base stations are connected to one another to form a wired point-to-point backbone network which handles all the ongoing activities in the cell, and has the overall control of the system resources. Signals from the base station are broadcast to all mobile subscribers. Signals sent from mobile subscribers are, however, only received by the base station and not heard by other mobile subscribers.

Each base station is allocated a number of frequencies (termed as channels or links) on which transmission takes places. At any time instant, only one station/subscriber can transmit on a channel; otherwise, collision occurs and all the ongoing transmissions on that channel fail. The channel used by the base station to transport data to mobile subscribers is called the *forward* channel, while that used by mobile subscribers to transport data to the base station is called the *reverse* channel. Signals on different forward/reverse channels are independent of one another. In the current OSU narrow-band wireless modem testbed, there are one forward channel and one reverse channel in each cell.

8.2.3 Physical characteristics of forward/reverse channels

The actual symbol rate on the forward channel is currently 3200 channel symbols per second. With a quaternary PSK modulation (QPSK) transmitting two coded bits per channel symbol, the forward channel can operate up to 6.4 kbps. The reverse channel supports approximately half the transmission rate of the forward channel in order to maintain a margin of path balance and allow time division multiple access. The actual symbol rate on the reverse channel is currently 2400 symbols per second. With the use of a PSK modulation transmitting two information bits per channel symbol, the reverse link can operate at a rate of 4.8 kbps.

Structure of pilot symbol frame: Transmission on both the forward/backward channels is broken into cycles of approximately 4 seconds in length. (The reason for choosing this cycle length will be given later.) Each notification cycle is further broken up into pilot symbol (PS) frames which have a size of 150 channel symbols. The structure of the PS frames is shown in Fig. 8.1 and described below: Fifteen pilot symbols are inserted every 10 symbols starting at the first symbol of the PS frame. In addition, 7 additional pilots are inserted at the beginning of each PS frame for a total of 22 pilot symbols per PS frame. The pilot symbols are used by the mobile unit to estimate the fading distortion and the resulting frequency offset. Each PS frame contains 128 channel symbols for the coded information bits, and hence the transmission efficiency of each PS frame is $128/150 = 85.3\%$.

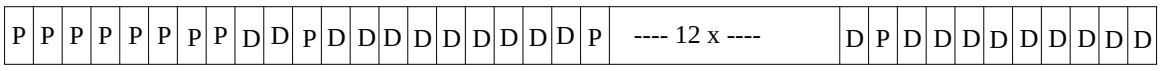


Figure 8.1: Pilot symbol frame, P=pilot symbol, D=data symbol

Each regular (non-real-time) data packet is 64 bytes (512 bits) long, 48 bytes (384 bits) of which are information bits. Since a PS frame contains 128 non-PS symbols (256 bits), each data packet is transmitted in two PS frames.

Error correction with Reed-Solomon code: For the purpose of error correction, packets transported on forward/reverse channels are encoded in Reed-Solomon code, RS(48,64), over GF(256). Our experience with wireless errors from field tests for this RS code design indicates that two events occur with extremely high probability: (i) a small number of errors occur and are corrected; and (ii) a large number of errors occur and the RS decoder fails to provide an output. Consequently it is extremely rare that a packet is delivered with an error. It is either delivered error free or the delivery fails (the latter is considered a packet loss by the base station and mobile subscribers because of lack of acknowledgments).

Insertion of preambles and guard times for synchronization: On the forward channel, a preamble is inserted at the beginning of each notification cycle to allow a mobile subscriber to synchronize to the master timing of the base station. The preamble will contain a unique word to synchronize the forward and reverse channels once every notification cycle. On the reverse channel, each non-real-time data packet is proceeded by a packet preamble of 600 symbols, followed by the packet body and a packet postamble of 51 symbols. Non-real-time packets are also separated from each other by a guard time of 0.0075 second (18 symbols). On the other hand, each GPS data packet is proceeded by a packet preamble of 64 symbols and separated from each other by a guard time of 0.0075 second (18 symbols). Table 8.1 summarizes these time parameters.

Packet size: On the forward channel, since each regular data packet is transmitted in two PS frames, it takes $300/3200 = 0.094$ second to transmit a data packet. On the reverse channel, it takes $300/2400 = 0.125$ second to transmit a non-GPS data packet. Together with the time to transmit the preamble and postamble ($651/2400 = 0.27125$ second) and the guard time ($18/2400 = 0.0075$ second), each data slot for transmission of non-GPS packets is set to 0.40375 second. On the other hand, it takes $128/2400 = 0.05333$ second to transmit a GPS packet. Together with the time to transmit the preamble ($64/2400 = 0.02667$ second) and the guard time ($18/2400 = 0.0075$ second), each data slot for transmission of GPS packets is set to 0.0875 second. Table 8.1 lists the parameters that characterize the physical layer characteristic and pertain to the MAC protocol design.

8.2.4 Constraints imposed by physical layer characteristics:

Half duplex transmission constraint: In the current narrow-band wireless testbed, the base station has a transmitter and a receiver, and can listen and transmit at the same time. However, because of the power and transmitter/receiver constraints, mobile subscribers can only transmit or receive but cannot do both at the same time. Moreover, a 20 ms guard time has to be inserted between switch-over from the transmit function to the receive function and vice versa. This implies that (i) a mobile subscriber cannot transmit 20 ms before or after its receiving period, and (ii) slots that carry packets destined to a mobile subscriber M on the forward channel must be apart from those scheduled to transport M's packets on the reverse channel by at least 20 ms. This is termed as the *half duplex transmission* constraint. Several important design decisions of the proposed MAC protocol have been driven by this physical layer constraint.

Asymmetry between the forward/reverse channels: Another physical layer characteristic that drives the design decision is the asymmetry between the forward and reverse channels. First, the base station can transmit with stronger power than mobile subscribers, and hence the forward channel is usually more reliable than the reverse channel. As a result, data packets transmitted on the reverse channel have to be proceeded by packet preambles and separated with guard time. Second, because of the physical layer characteristics (e.g., difference in the symbol transmission rate and the modulation schemes, and the necessity of packet preamble/postamble/guard time on the reverse channel), the reverse channel has comparatively much less bandwidth than the forward channel. Third, as a result of the half duplex transmission constraint, mobile subscribers cannot be scheduled to transmit and receive at the same time, and proper guard times have to be inserted between the switch-over of transmit and receive functions. Finally, without a centralized control facility, mobile subscribers may compete for channel access on the reverse channel, sometimes on a contention basis, and hence the access delay on the reverse channel is longer and more variable.

8.3 Proposed MAC Protocol – OSU-MAC

8.3.1 Base station-centric resource arbitration

A major feature of our proposed MAC protocol is that we delegate to the base station the full responsibility of resource arbitration, channel access, and registration in each cell. The reasons for this design are two-fold: first, because the base station in a cell usually has the overall control over all the system resources, it is reasonable to delegate the base station to arbitrate the assignment of data slots on both the forward and reverse channels. Second, because of the asymmetry between the forward and reserve channels, the MAC protocol should be so designed as to include as little control overhead on the reverse channel as

possible. That is, the control information sent from mobile subscribers to the base station should be kept minimal. This leads to a base station-centric mechanism. The only control information sent uplink is the registration and slot reservation requests.

Specifically, the base station transports data packets as well as channel access information on the forward channel to mobile subscribers. Channel access information includes, among other things,

- the slot access schedule on the forward channel and on the reverse channel for the current notification cycle. In particular, the slot access schedule on the reverse channel is determined by the reservation requests received on the reverse channel in the previous notification cycle.
- acknowledgment for packets received by the base station on the reverse channel.
- information used to page inactive mobile subscribers.

Control fields: Each mobile subscriber has a permanent, universally unique equipment identification number (EIN) of 16 bits. In addition, a mobile subscriber is assigned a user ID of 6 bits when it registers with the base station. This 6-bit user ID is unique only within the cell, and will be used solely by the base station to specify/identify a mobile subscriber. A set of explicit control fields on the forward channel is used for the base station to convey the above channel access information to mobile subscribers. There is no explicit control field on the reverse channel. All the control information sent uplink is either carried in the header of data packets or included in regular data packets (i.e., the in-band signaling approach is used). The control fields consist of the following information (Fig. 8.2):

- GPS schedule: gives the user IDs of (up to) 8 GPS users which are scheduled to use the 8 GPS slots on the reverse channel. This field is $8 \times 6 = 48$ bits.

GPS schedule	Reverse schedule	Forward schedule	Reverse ACKs	Paging
48 bits	54 bits	222 bits (for 37 forward data slots)	198 bits (for 9 reverse data slots)	108 bits

	Forward channel
GPS schedule (bits)	48
Reverse schedule (bits)	54
Forward schedule (bits)	222
Reverse ACKs (bits)	198
Paging (bits)	108
Total size of control fields (bits)	630
Required RS codewords for one set of control fields	2
Required PS frames for one set of control fields	4
Required time for one set of control fields (seconds)	0.1875

Figure 8.2: The control fields on the forward channel

- Reverse schedule: is used by the base station to announce the slot schedule in response to the reservation requests received on the reverse channel, either *explicitly* or *implicitly*[5], in the previous notification cycle. It gives the user IDs of (up to) M data users scheduled to use the data slots on the reverse channel. In our current design[6] $M = 9$, and hence this field is $9 \times 6 = 54$ bits.

- Forward schedule: is used to inform mobile subscribers to which subscriber data slots on the forward channel will be transmitted. It gives the user IDs of mobile subscribers which should receive data on the N data slots on the forward channel. In our current design [7] $N = 37$, and hence this field is $37 \times 6 = 222$ bits.

[5]To be discussed below.

[6]The reason why $M = 9$ will be given in Section 8.3.3.

[7]The reason why $N = 37$ will be given below.

- Reverse ACKs: are used to acknowledge receipt of data packets on the reverse channel in the previous notification cycle or to notify a mobile subscriber (whose registration request is approved) of its (EIN, user ID) pair. This field is $9 \times (16+6) = 198$ bits.

- Paging: is used to page and locate inactive mobile subscribers. To support paging of up to 18 users, this field contains $18 \times 6 = 108$ bits.

The total length of these control fields is 630 bits, which requires 2 RS codewords to carry. Note that out of the 768 informational bits available in the 2 RS codewords, 138 bits are reserved for future use. All mobile subscribers have to listen to the control fields on the forward channel in order to find out their scheduled access time to the channels.

Slot reservation and scheduling: For real-time, GPS applications transported in uplink, we use reservation-based scheduling to ensure the QoS required. When a mobile subscriber using GPS applications registers with the base station, it will be assigned GPS slots properly spaced on the reverse channel until the mobile subscriber signs off. The GPS slots will be so assigned that at least one GPS slot in any time interval of 4 seconds is assigned to the GPS subscriber.

To allow mobile subscribers with regular, non-real-time data to gain access to the reverse channel, one or more data slots on the reverse channel are designated as *contention* slots in each notification cycle, where a contention slot is simply a data slot *not* assigned to any mobile subscribers on the reverse channel. There are three possible means to reserve data slots on the reverse channel:

1. A mobile subscriber may explicitly send a reservation request packet, specifying the number of data slots desired, on one of the contention slots.

2. When a mobile subscriber transmits its data packets in the data slots assigned to it in a notification cycle, it may set a reservation field in the header of the data packet to implicitly indicate that it would like to request more data slots in the next notification cycle.

3. A mobile subscriber may send its data packet in one of the contention slots and compete with the other mobile subscribers with data or reservation packets on a contention basis. If multiple mobile subscribers attempt to transmit their data/reservation packets in the same slot, collision occurs. (The base station has to explicitly acknowledge on the forward channel receipt of data packets on the reverse channel as a result of these potential collisions in contention slots.)

When collision occurs, mobile subscribers back off with a random period of time before their subsequent attempts. (To increase the probability of successful reservation, mobile subscribers that transmit data packets without reservation are required to back off with a longer time period.) Also, if collisions occur multiple times in a notification cycle or across multiple notification cycles, the base station may designate additional data slots as contention slots (i.e., leave them unassigned) in the next cycle. On the other hand, if multiple contention slots have been left unused in the current cycle, the base station may decrease the number of contention slots in the next cycle.

The base station notifies a mobile subscriber that makes a reservation request or transmits its data in a contention slot of whether or not the request/data has been received by indicating in the ith reverse ACKs field the user ID of the subscriber whose request/data has been received. Note that the fact that a reservation request is received does not imply the request will be honored. A mobile subscriber has to look into the reserve schedule field to find out whether or not it is assigned data slots.

After the base station "collects" all the requests in the current notification cycle, it uses a specific scheduling algorithm (in our current design, the round robin algorithm) to determine the slot schedule on the reverse channel, subject to the half-duplex transmission constraints. The resulting slot schedule is then announced in the reverse schedule control field in the next notification cycle. Mobile subscribers will then transmit their data accordingly.

8.3.2 Registration of mobile subscribers

A mobile subscriber registers itself with the base station through the use of contention slots. A mobile subscriber that newly enters the cell first listens to the forward channel to synchronize itself on the reverse channel and to find out the positions of contention slots. Then it transmits its registration request in one of the contention slots. The registration request packet may compete with other registration/reservation/data packets. If a collision results, the mobile subscriber with the registration request *persists* in the next notification cycle until it succeeds in one notification cycle or fails after a pre-determined number of attempts. Note that we give the priority of using contention slots to mobile subscribers attempting to register themselves with the base station as other mobile subscribers with reservation/data packets will back off in the case of collision.

If a registration request made in the ith contention slot is successfully received by the base station, it will be passed to the registration handling module for approval. If the registration request is approved, the base station will notify the requesting subscriber by returning the (EIN, user ID) pair in the ith reverse ACKs field.

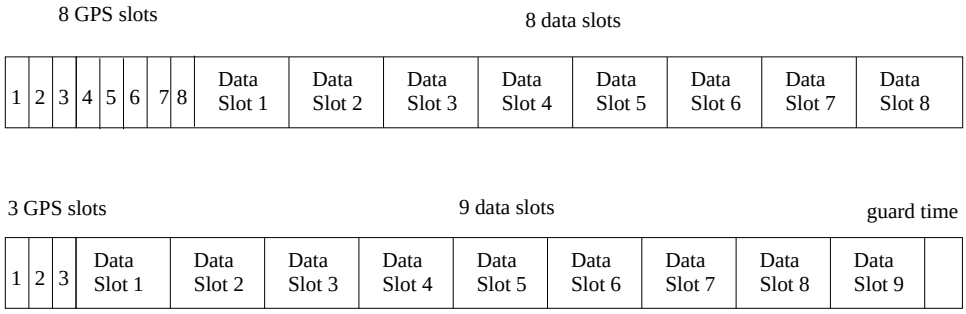


Figure 8.3: Two formats of the notification cycle on the reverse channel

8.3.3 Structure of notification cycle on the reverse channel

The structure of the notification cycle on the reverse channel is shown in Fig. 8.3. Depending on the number of active GPS subscribers, the system can choose one of the two possible formats. If there are more than three active GPS subscribers, the system uses the first format. In this format, 8 GPS slots are scheduled first, followed by 8 data slots. The second format is used when the number of active GPS subscribers is less than or equal to 3. In this case, five unused GPS slots are combined to form a data slot (to be used by data users). The notification cycle begins with 3 GPS slots, followed by 9 data slots and a guard time of 0.03375 second.

Since the base station knows how many active GPS subscribers are in the system, it is the base station's responsibility to choose and announce which format to use. The announcement is made implicitly through the number of GPS subscribers in the control fields. If the number is greater than 3, then the base station and all mobile subscribers will use format 1, otherwise, they use format 2. By using a format, we mean the mobile subscribers will synchronize themselves to the beginning of each notification cycle and access the reverse channel according to the time given in Table 8.2.

Dynamic GPS slot adjustment on the reverse channel: As mentioned in Section 8.2.1, GPS units report the bus location once every 4 seconds. A naive approach to fulfill this real-time requirement is to statically allocate the same GPS slot in each notification cycle to a registered GPS user. This approach, although simple, may result in bandwidth waste. This is because as GPS users register and later sign-off, some of the GPS slots may be allocated and then released, creating holes between allocated GPS slots. For example, if GPS users 1 to 8 registered and assigned GPS slots in order. Later, users 2, 3, 5, 6 and 7 left the system, creating two holes slots 2–3 and slots 57. These holes cannot be used by data users, even if the number of GPS users is less than 3.

A more sophisticated approach is to dynamically adjust GPS slots to consolidate allocated GPS slots and then combine unused GPS slots into data slots. If there are more than three GPS users, the system uses format 1. When GPS users leave, GPS slots are re-assigned to existing GPS users and unused GPS slots converted into a data slot, *subject to the real-time requirements of existing GPS users*. If more GPS users register later, this data slot can be split into five GPS slots again. The real-time requirements of existing GPS users are ensured through the following rules of slot re-assignment:

(R1) The GPS slots in a cycle are allocated in order.

(R2) When a GPS user is admitted into the system, it is allocated the first unused GPS slot.

(R3) When a GPS user assigned GPS slot i leaves the system, the GPS user that uses GPS slot $j > i$ (if any) is re-assigned slot i.

Note that with (**R3**), when a GPS user is re-assigned a slot, it is ensured to have an slot access interval that is less than 4 seconds in the current notification cycle and hence the real-time requirement is fulfilled. Also, with these rules, allocated GPS slots are consolidated

at the beginning of each notification cycle, and unused GPS slots can be converted into a data slot.

The number of regular data slots on the reverse channel: Given that (1) each notification cycle is approximately 4 seconds long, (2) there are at most 8 GPS slots, each of length 0.0875 second, and (3) each non-real-time data slot is 0.40375 second in length (Table 8.1), the total number of regular data slots is

$$(4 - 0.0875 \times 8)/0.40375 \approx 8 \tag{8.1}$$

Guard time: The exact cycle length on the reverse channel under the above configuration is 3.93 seconds. As will be discussed in Section 8.3.4, the notification cycle length on the forward channel is 3.9844 seconds. To make the notification cycles on both channels the same, we add a guard time of 0.0544 second on the reverse channel.

8.3.4 Structure of notification cycle on the forward channel

The notification cycle on the forward channel begins with a preamble of 300 symbols. The preamble is then followed by control fields and data slots.

Two control fields to deal with the half-duplex transmission constraint: As mentioned above, the base station uses the control fields on the forward channel to announce channel access schedules on both channels, to acknowledge packets received on the reverse channel, and to page inactive mobile subscribers. All mobile subscribers must listen to the control fields to obtain the above information. However, due to the half-duplex transmission constraint imposed by the physical layer, mobile subscribers scheduled to transmit on the reverse channel at the same time of the control fields being transmitted on the forward

channel will not be able to listen to the control fields. One straightforward solution is not to schedule any mobile subscriber for transmission on the reverse channel during the period when the control fields are being transmitted on the forward channel. We did not adopt this solution, because it results in a waste of bandwidth on the reverse channel.

As an alternative to deal with the half-duplex transmission constraint, we insert two sets of control fields, with the second one exclusively used by mobile subscribers scheduled to transmit during the time interval when the first control fields are transmitted. In other words, the bandwidth utilization on the reverse channel is improved at the expense of introducing a second set of control fields on the forward channel (which has comparatively more abundant bandwidth).

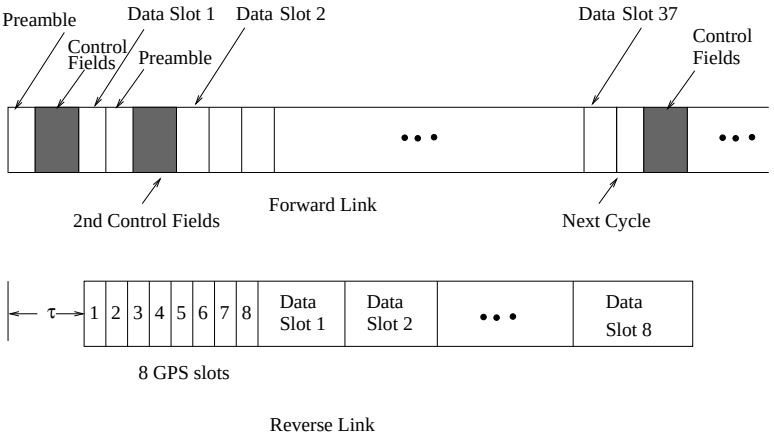


Figure 8.4: The structure of notification cycles on the forward and reverse channels

There are three issues that must be resolved before we can fully realize the two control field design:

The location of the second set of control fields: One intuitive approach is to evenly divide data slots into two groups and arrange the preamble, control fields, and data slots in each notification cycle as follows:

Preamble	Control fields	Half of the data slots	Preamble	Control fields	Half of the data slots

The problem with the above arrangement is that the base station cannot use the first half of data slots on the forward channel to send data to the mobile subscribers that listen to the second set of control fields. This is because these subscribers will not know their schedule until they listen to the second set of control fields. Similarly, the reverse data slots that are ahead (in time) of the second set of control fields cannot be assigned to these mobile subscribers either. To deal with this problem, we propose to place the second set of control fields as close to the beginning of each notification cycle as possible. As shown in Fig. 8.4, the notification cycle is structured as the preamble (of size 300 symbols) followed by the first set of control fields (2 RS codewords), one data slot (1 RS codeword), another preamble (of size 150 symbols), the second set of control fields (2 RS codewords), and the rest of data slots. With this configuration, in the worst case (which occurs when the base station only has packets destined for the data user which is scheduled to transmit in the data slot), only one data slot on the forward channel cannot be used.

The reason for not concatenating the two sets of control fields back to back is to ensure that the data user scheduled to transmit in the last reverse data slot completes its uplink transmission and has sufficient time to switch from the transmit function to the receive function.

The set of control fields a mobile subscriber should listen to: All mobile hosts need to listen to the control fields in order to synchronize with the base station and to receive the channel access schedule. With the two control field design, some users will transmit at the same time as the control fields. How do they know which control field they should listen to? In order to solve this problem, we shift the cycle on the reverse channel τ seconds later than that on the forward channel (Fig. 8.4), where

$$\tau = 0.09375(\text{preamble}) + 0.1875(\text{control fields}) + 0.02 = 0.30125 \text{ seconds} \quad (8.2)$$

The extra 0.02 seconds makes it possible for the GPS users to transmit right after they learn their schedules on the forward channel.

After the shift, the only slot on the reverse channel that overlaps the first control fields in the next notification cycle is the last data slot. The user which is scheduled to transmit in this slot should listen to the second set of control fields. All the other users should listen to the first set of control fields. In summary, mobile subscribers use the following rules to decide which set of control fields they should listen to: (i) When a mobile subscriber first enters the system, it listens to the first control fields; and (ii) If a mobile subscriber is assigned to transmit in the last reverse data slot, it listens to the second set of control fields; otherwise, it listens to the first set of control fields.

Note that the base station must not assign the first slot on the forward channel to the user which listens to the second set of control fields.

Difference between the first and second sets of control fields? The only difference between the two sets of control fields is that the second set of control fields has to

acknowledge the activity that occurs on the reverse channel when the first set of control fields is transmitted. Specifically,

- If the last data slot on the reverse channel was used to send a data packet, the second set of control fields acknowledges its reception (in the reverse ACKs field).
- If the last data slot on the reverse channel was used by a new mobile subscriber for registration, the second set of control fields announces whether or not the reservation succeeds.

Also, the base station can schedule forward data slots that were announced idle in the first set of control fields to the user assigned the last data slot on the reverse channel, based on whether or not the user requests more data slots in the packet header of its packet (transmitted in the last slot). However, the base station *cannot* make any change in the reverse schedule.

The number of data slots per cycle on the forward channel: The number of data slots that can be transmitted per notification cycle is contingent, among other things, upon the sizes of data slots and notification cycle. Since each data packet is 2 RS codewords long, we decide that each data slot is of size 2 RS codewords (300 symbols with both pilot symbols and Reed-Solomon error check bits considered) as well. Given that (1) the forward channel can transmit 12800 symbols in a 4-second period, (2) the two preambles total 450 symbols, (3) the two sets of control fields are 600 symbols (2 RS codewords) each, and (4) each data slot is of size 300 symbols (Table 8.1 and Fig. 8.2), the number of data slots available is thus $(3200 \times 4 - 450 - 600 \times 2)/300 \approx 37$. This implies that the exact length of a notification cycle is 3.9844 seconds.

8.3.5 Scheduling constraints and algorithm

As mentioned in Section 8.3.1, we delegate to the base station the full responsibility of (i) generating slot schedules on both the forward and reverse channels and (ii) handling registration requests in each cell. After introducing the notification cycle structure on both the forward and reverse channels, we are now in a position to delve into the details of how data slots are scheduled on both channels. Since the reverse channel has a more limited bandwidth than the forward channel, the base station schedules slots on the reverse channel and then assigns slots on the forward channel, the latter subject to the half-duplex-transmission and two-control-fields constraints.

Generation of slot schedules for data/GPS applications: Mobile subscribers make their reservation requests for the reverse channel through explicit/implicit reservation or contention. The slot schedule for regular data slots on the reverse channel is then generated using the round robin scheduling algorithm. After the schedule is determined by the scheduling algorithm, the schedule is then re-adjusted to lump slots allocated to a mobile subscriber together so that the subscriber does not have to repeatedly switch between transmitting and sending in a cycle.

After the reverse slots are scheduled, the data slots on the forward channel are allocated in a similar way, but subject to the following constraints: (i) a mobile subscriber cannot be scheduled to transmit on the reverse channel and to receive on the forward channel at the same time; (ii) a mobile subscriber cannot be scheduled to transmit on the reverse channel 20 ms before or after it is scheduled to receive on the forward channel; and (iii) a mobile subscriber cannot be scheduled to receive on the forward channel 20 ms before or after it is scheduled to transmit on the reverse channel.

Registration and reservation: In order to allow new users to register and registered users to send reservation requests, the first few data slots in a notification cycle are always left unassigned and used as contention slots. Users can use contention slots to register themselves with the base station or make slot reservation.

In order to reduce the registration latency (defined as the time interval between the time when a registration is first made and the time it is finally received by the base station), the base station monitors the collision rate in the contention slots. If the rate exceeds some predetermined threshold, the base station dynamically allocates a few more contention slots in the subsequent notification cycles, and vice versa.

8.4 Performance Evaluation

We have implemented OSU-MAC in a Java-based simulation environment, called *JavaSim* [72], and conducted a simulation study to validate the proposed design. We do not include a simulation comparison to the other existing protocols (summarized in Section 2.6) because all the protocols have been designed with different objectives under different environments: OSU-MAC has been designed for both bus tracking applications and regular data applications on a specific testbed, while other protocols were designed either for voice/data users (PRMA, D-TDMA, RAMA, DRMA, FAMA), for real-time/best-effort traffic on an abstract model (RQMA), or for cable modem users (MCNS). A comparison among them would not be fair.

The simulation scenario is as follows: there are up to 8 buses within the cell covered by a base station. Each bus carries a GPS unit that transmits GPS packets periodically to report its location. Also, mobile subscribers in the cell may send/receive short e-mails on the reverse/forward channel. For the purpose of evaluating the MAC performance, we

assume the e-mail messages are generated at a mobile subscriber according to a Poisson process with mean inter-arrival time T. Two types of packets are used in the simulation: packets of fixed length $L = 120$ bytes and variable-length packets whose length is drawn from a uniform distribution between 40 and 500 bytes. When the number of GPS users is less than or equal to (greater than) 3, each notification cycle on the reverse channel has $d = 9$ ($d = 8$) data slots, among which the first is a contention slot for registration and reservation.

Given that there are m mobile data subscribers in the cell, the *load* index ℓ of the reverse channel is defined as

$$\ell = \frac{\frac{m \times 3.9844}{T} \times L}{40 \times d} \tag{8.3}$$

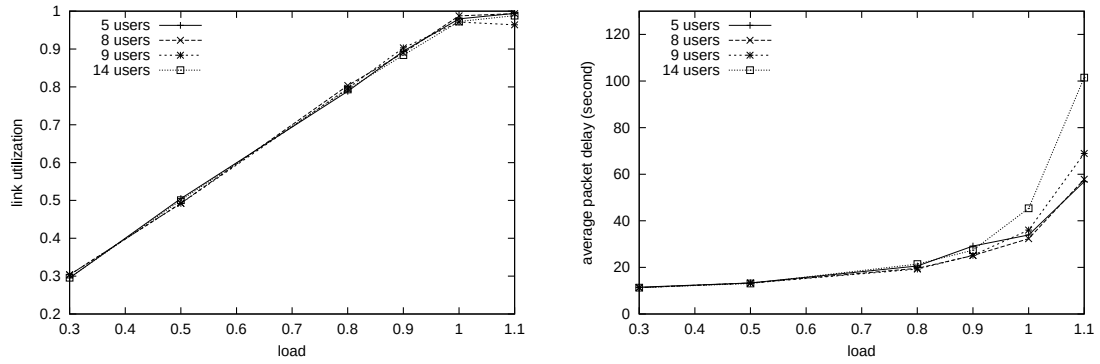
where $\frac{m \times 3.9844}{T}$ is the average total number of messages generated in a notification cycle, $\frac{m \times 3.9844}{T} \times L$ is the total number of bytes generated, and $40 \times d$ is the total number of data bytes that can be transported in the d data slots on the reverse channel.

The simulations are designed to evaluate the system performance under light, medium and heavy loads, with the value of ℓ varying from 0.3, 0.5, 0.8, 0.9, 1.0 to 1.1, the number of GPS users varying from 1 to 8, and the number of data users varying from 5 to 14. Given different combinations of traffic conditions, the inter-arrival time T is calculated as

$$T = \frac{m \times L \times 3.9844}{40 \times d \times \ell} \tag{8.4}$$

In spite of several system parameters involved, the results are found to be quite robust in the sense that the conclusion drawn from the performance curves (reported below for the variable-length packet case) is valid over a wide range of parameter values.

Utilization on the reverse channel: The link utilization, defined as the percentage of the available bandwidth used to carry data on the reverse channel, versus the load index ℓ is



(a) Link utilization (b) Packet delay

Figure 8.5: Link utilization and packet delay

shown in Fig. 8.5 (a). When the load index $\ell \leq 0.8$, most packets get through, and the link utilization is close to the traffic load. When the load index is close to 1, some packets are dropped because of buffer overflow, and the link utilization is smaller than the traffic load.

Packet delay: Packet delay versus the load index is depicted in Fig. 8.5 (b). When $\ell \leq 0.5$, packets can be delivered in three to five cycles, even in the case of variable-length packets (with an average packet size of 280 bytes). This, coupled with the fact that the bandwidth available on the reverse channel is limited due to the physical layer characteristics, demonstrates the ability of OSU-MAC to accommodate a large number of mobile subscribers, while maintaining high utilization and small packet delay under small to medium loads. When the load increases beyond 0.9, the packet delay increases dramatically, due to the fact that the traffic load grows beyond the system capacity and packets start to queue up.

Control overhead: We use the ratio of the number of reservation packets (transmitted in contention slots) to the total number of data packets (transmitted in data slots) as an index of control overhead. As depicted in Fig. 8.7, counter-intuitively the control overhead decreases as the load increases. This is because as the load increases, reservation requests are usually piggybacked in the reservation bit of the packets sent uplink, leading to the smaller number of reservation packets. Due to the same reason, as the load increases, the probability that collision occurs in contention slots decreases (Fig. 8.6 (a)), and the average reservation latency also decreases (Fig. 8.6 (b)).

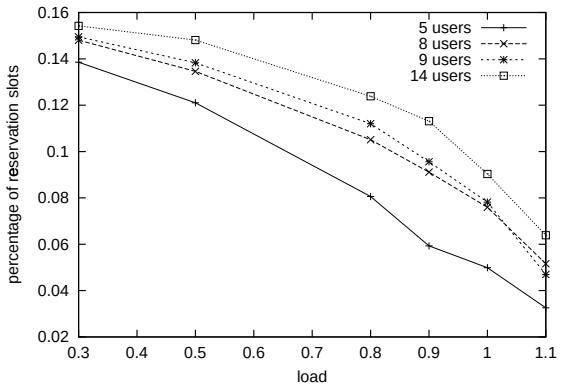
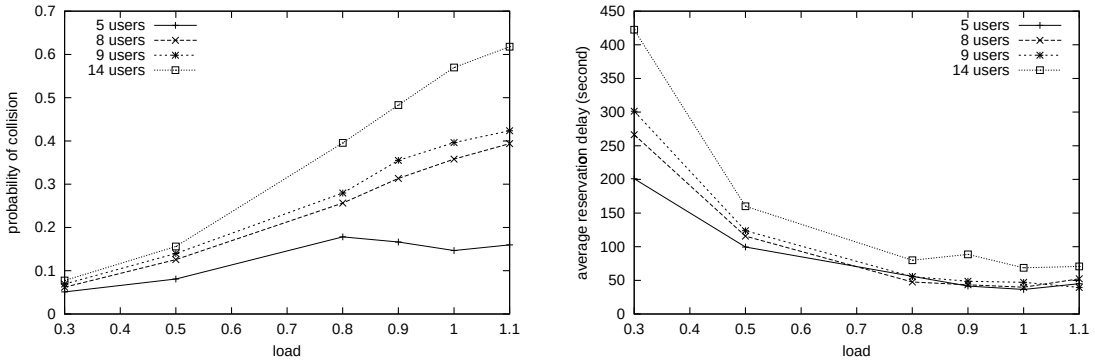


Figure 8.6: Control overhead as a function of load

Fairness: As described in Section 8.3.5, we use the round robin algorithm to assign data slots on the reverse channel to mobile subscribers with data packets. As shown in Fig. 8.8, OSU-MAC ensures fairness among mobile subscribers (i.e., the fairness index under all traffic loads are over 0.99), where the fairness index is defined as [43]

$$\frac{n \cdot \sum_{i=1}^{n} u_i^2}{(\sum_{i=1}^{n} u_i)^2}$$



(a) The probability of collision (b) Reservation latency

Figure 8.7: Probability of collision in contention slots and the reservation latency

and u_i is the bandwidth the i mobile subscriber acquires.

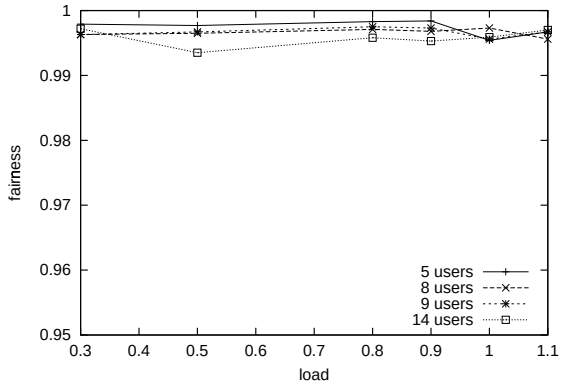
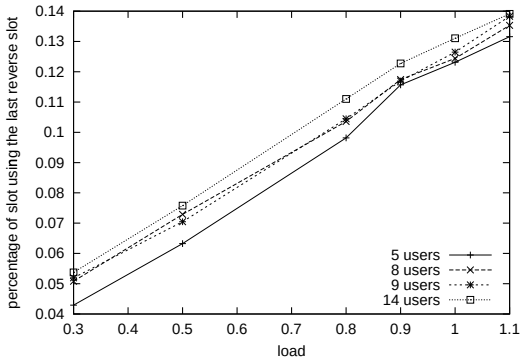
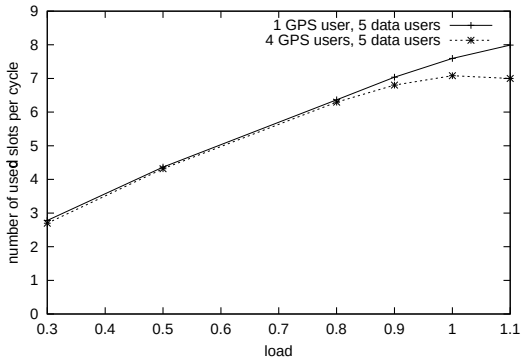


Figure 8.8: Fairness

Performance improvement due to the two-control-fields design and dynamic slot adjustment: Fig. 8.9 (a) gives the percentage of bandwidth gain by using the second set of control fields. This is obtained by calculating the ratio of the number of data packets sent



(a) by two control fields.



(b) by dynamic slot adjustment.

Figure 8.9: Performance improvement by two control fields and dynamic slot adjustment

in the last data slot on the reverse channel to the total number of data packets sent (as the last data slot on the reverse channel overlaps with the first set of control fields). As little as 5% and as much as 14% of the bandwidth is saved by use of the second set of control fields.

Fig. 8.9 (b) depicts the average number of data slots that have been used in the case that the number of GPS users is 1 and 4, respectively. Recall that when there are 3 or less GPS users, 5 GPS slots will be converted to an additional data slot. Hence, Fig. 8.9 (b) shows how effective the dynamic slot re-adjustment approach helps in utilizing bandwidth originally allocated to unused GPS slots. The effect of dynamic slot re-adjustment is not significant when the load is light, but as much as 15% more bandwidth can be utilized with slot re-adjustment.

8.5 Chapter Summary

In this chapter, we have designed and implemented a MAC protocol, *OSU-MAC*, that coordinates the transmission activities on the forward and reverse channels so as to support

both the real-time bus location tracking application and the regular data applications, on an OSU narrow-band wireless modem testbed. There are several unique features of OSU-MAC: first, we delegate to the base station the full responsibility of resource arbitration, channel access, and registration in each cell. This base-station-based scheduling approach enables provisioning of deterministic QoS for real-time applications, effective supports of slot reservation and mobile registration, while maximizing system utilization. Second, we take into account the error characteristics of the physical layer. In particular, we consider the half-duplex transmission constraint and propose a two-control-field structure to fully utilize the limited bandwidth available on the reverse channel. Third, to make use of the unused bandwidth originally reserved for real-time traffic, we also propose a dynamic slot adjustment scheme. Finally, as a real MAC protocol currently being implemented on the OSU narrow- band wireless modem testbed, we insert, wherever needed as advised by the wireless modem researcher, preambles, postambles, and guard times between packet slots or notification cycles for synchronization. We are currently implementing OSU-MAC on the MS-Windows operating system and will develop a real-time bus tracking application and an email delivery system to demonstrate the use of *OSU-MAC* for distributed applications.

	Fwd channel	Rev channel
General physical layer characteristics		
Chan. symbol rate (symbols per second)	3200	2400
Coding rate (coded bits/symbol)	2	2
Information symbols in a pilot frame	128	128
Chan. symbols in a pilot frame	150	150
Information bits per RS (64,48) codeword	384	384
Bits per RS (64,48) codeword	512	512
Packet size		
RS codewords per packet	1	1
Pilot frames per regular data packet	2	2
Chan. symbols per regular packet	300	300
Time per regular packet (second)	0.09375	0.125
Cycle preamble		
Cycle preamble length (chan. symbols)	450	n/a
Time per cycle preamble (seconds)	0.140625	n/a
Packet parameters on reverse channel		
	GPS	Regular
Packet size (information bits)	72	384
Packet size (chan. symbols)	128	300
Packet preamble (chan. symbols)	64	600
Packet preamble (seconds)	0.02667	0.25
Packet postamble (chan. symbols)	0	51
Packet postamble (seconds)	0	0.02125
Packet guard time (chan. symbols)	18	18
Packet guard time (seconds)	0.0075	0.0075
Total length (chan. symbols)	210	969
Total length (seconds)	0.0875	0.40375

Table 8.1: List of parameters in the physical layer that pertain to the MAC design

	Format 1	**Format 2**
GPS slot 1	0.30125	0.30125
GPS slot 2	0.38875	0.38875
GPS slot 3	0.47625	0.47625
GPS slot 4	0.56375	—
GPS slot 5	0.65125	—
GPS slot 6	0.73875	—
GPS slot 7	0.82625	—
GPS slot 8	0.91375	—
Data slot 1	1.00125	0.56375
Data slot 2	1.40500	0.96750
Data slot 3	1.80875	1.37125
Data slot 4	2.21250	1.77500
Data slot 5	2.61625	2.17875
Data slot 6	3.02000	2.58250
Data slot 7	3.42375	2.98625
Data slot 8	3.8275	2.98625
Data slot 9	—	3.39000

Table 8.2: Reverse channel access time of the two formats

CHAPTER 9

PACKING DENSITY OF VOICE TRUNKING USING AAL2

ATM Adaptation Layer Type 2 has been adopted in ITU-T and ATM Forum to reduce the packing delay for voice trunking. A parameter called "Timer_CU" is used in AAL2 to avoid prolonged delays for voice packets. However, the AAL2 documents do not discuss how to set the Timer_CU value. In this chapter, we analyze the tradeoff between delay and bandwidth efficiency, and establish a guideline for setting the Timer_CU value.

9.1 Introduction

Since the emergence of computer networks, efforts have been made to transfer voice over networks[21, 20, 56]. Starting as a technical novelty, Internet telephony is now becoming a big business. However, the quality of Internet phone still remains a problem. Because telephony is a real-time application, delay, among other quality measurements, is the most important factor that affects the quality of voice. If a packet arrives late, its contents become obsolete and have to be discarded. According to ITU-T Recommendation G.114[37], an end-to-end delay of 0 to 150 ms is acceptable for most user applications. A delay of 150 to 400 ms is acceptable provided that administrators are aware of the transmission time impact on the transmission quality of user applications, but any delay above 400 ms is unacceptable for general network planning purposes.

The problem of delay becomes more severe when efficient compression/decompression methods are used. For example, in order to fill an ATM cell which has 48-byte payload, the ITU-T G.711 (64 kbps) codec needs 6 ms, but the more efficient ITU-T G.723.1 (5.3 kbps) codec needs 72 ms. Notice this 72 ms does not include the propagation delay, queueing delay, etc, that the cell must undergo when it travels through the networks.

ATM Adaptation Layer 2 (AAL2) has been designed to reduce the packing delay. It is described in ITU-T Recommendation I.366.2[38] and ATM Forum specification "ATM Trunking using AAL2 for Narrowband Services"[4] . The idea is to multiplex voice packets from several sources into one ATM cell so that the time to fill a cell can be reduced significantly. Figure 9.1 illustrates a scenario of voice packets from three sources being packed into cells.

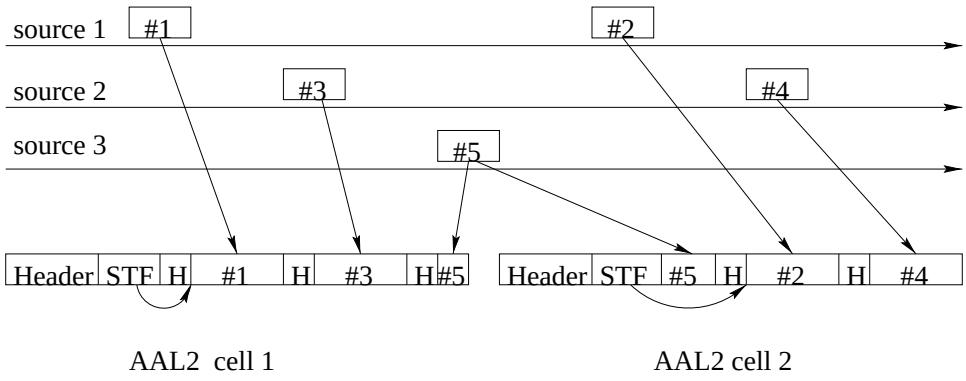


Figure 9.1: AAL2 cell packing

Also shown in Figure 9.1 is the format of AAL2 cells. Every cell has a standard 5-byte ATM header. Following the header is the Start Field (STF) that indicates where in the payload the next complete packet starts. For every packet, there is a 3-byte mini-header

(H) which includes the Channel ID (CID), Length Indicator (LI), User-to-User Indication (UUI) and Header Error Control (HEC) of the packet.

However, the above packing mechanism has one problem. Suppose the first packet is put into a cell and is waiting for the arrival of other packets to complete the cell. But after an extended period, no other packets arrive. This packet will suffer a prolonged delay.

To avoid such prolonged delay, a parameter called "Timer_CU" is proposed in [38]. When the packing begins, a timer is set to this parameter value. If the cell is not completely packed within the time period determined by this Timer_CU value, the timer expires and the partially packed cell will be sent.

Zhang[77] analyzed the impact of Timer_CU value on packet delay variation (PDV), and found that the Timer_CU value is not relevant in the total PDV calculation. While the statement about PDV is correct, we found that the Timer_CU value has to be set appropriately since it significantly affects the link efficiency. If the Timer_CU value is too small, more partial cells are likely to be sent and the link efficiency will be low. If it is too large, some packets will suffer a prolonged delay and the voice quality will degrade.

In this chapter, we establish a Markov chain model to analyze the AAL2 packing process using ITU-T G.723.1 voice encoding. The Markov analysis reveals the correlation between successive cells and gives a formula for calculating the packing density based on the Timer_CU value and the number of voice sources in the system. To validate our analytic result, a simulation was launched to actually implement the AAL2 packing process. The results of the simulation perfectly match our analysis. The comparison is presented in section 5.

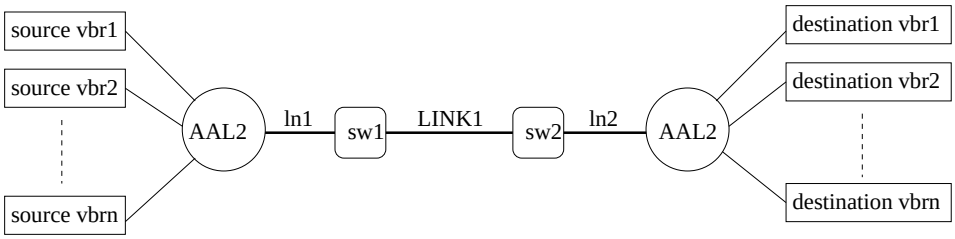


Figure 9.2: AAL2 simulation model

9.2 Simulation Model

Figure 9.2 shows our simulation model and the packing process. On the left, a number of variable bit rate (VBR) voice sources are connected to the source AAL2. The corresponding VBR destinations are shown on the right. Voice sources send voice signal in the form of packets to the source AAL2. The source AAL2 keeps a working cell. Arriving packets are packed into the working cell. Once the working cell is packed, it is sent to the switch.

When the first packet is put in the cell, a timer is set to Timer_CU parameter value. If the cell is not fully packed within Timer_CU time, the cell "*expires*" and is sent even though it is partially packed.

If an arriving packet cannot fit in the remaining space in the working cell, the bytes that can be fit in are put into the available space, and the cell is dispatched. The remainder of the packet is put in the next cell and the timer is initialized to Timer_CU.

The cells go through links and switches in the networks and arrive at the destination AAL2. The destination AAL2 unpacks the cells and dispatches each packet to its destination according to the channel identifier (CID) contained in the mini-header of the packet.

9.3 Voice Model and Packet Arrival Pattern

Human voice consists of alternating talkspurts and silence intervals. It has been found that talkspurt lengths and silence intervals are exponentially distributed[13]. In a commonly accepted model, the talkspurts have a mean length of 352 ms and silence intervals have a mean length of 650 ms [22].

There are a number of standards for coding voice. ITU-T G.723.1 is currently the most widely used encoding standard. During talkspurts, G.723.1 sends out a 20-byte packet every 30 ms. During silence periods, no packets are sent. In this chapter, we assume all voice sources use G.723.1 as the encoding method.

Compared with the talkspurt length and silence period, the 30 ms packet length is short. This means, during a talkspurt, the voice source emits a series of packets at 30 ms intervals. Therefore, the arrivals are highly predictable.

Suppose that the number of voice sources in the system is N and that a packet has just been received from a voice source. We want to calculate the probability of no packet arrival from all sources in the next τ ms, where τ is the Timer_CU value.

Consider one voice source first. During an average 352 ms talkspurt and 650 ms silence cycle, 12 packets need to be sent as shown in Figure 9.3.

Figure 9.3: Packet arrival pattern from one source

Let s denote the number of packets generated during a talkspurt interval. In order to have no packet arrival in τ ms, the starting point of this τ ms cannot fall in the τ ms interval before any of these s arrivals. The probability of no packet arrival from this source is

$$\frac{1002 - s\tau}{1002} \tag{9.1}$$

Here we assume $\tau < 30$. Since the average number of packets in a talkspurt is 12, the probability of no packet arrival from one source is

$$p = \frac{1002 - E[s]\tau}{1002} = \frac{1002 - 12\tau}{1002} \tag{9.2}$$

Voice sources are assumed to be independent. After the receipt of the first packet in the cell, the probability of no arrivals from all other sources within τ ms is

$$R_0 = p^{N-1} \tag{9.3}$$

and the probability of one packet arrival from all other sources is

$$R_1 = (N-1)p^{N-2}(1-p) \tag{9.4}$$

9.4 Calculation of Packing Density

Using the probabilities R_0, R_1, we can calculate the average packing density.

The average number of bytes in an AAL2 cell depends on whether there is a remainder from the last cell and how many packets are received since the first packet was put in the cell. Let r_n be the remainder length left from the $(n-1)$-th cell. This r_n, then, is exactly the STF field in the n-th AAL2 cell. Since each packet has a 3-byte header and a 20-byte payload, the remainder is always shorter than 23 bytes, i.e., $0 \leq r_n \leq 22$.

Event $r_n = 0$ happens only when cell $n-1$ expires or when $r_{n-1} = 1$ and cell $n-1$ does not expire;

Event $r_n = 1$ happens only when $r_{n-1} = 2$ and cell $n-1$ does not expire;

Event $r_n = 2$ happens only when $r_{n-1} = 3$ and cell $n-1$ does not expire;

.....

Event $r_n = 22$ happens only when $r_{n-1} = 0$ and cell $n-1$ does not expire.

Therefore, $\{r_n\}$ forms a Markov chain. Consider the stationary state where all $\{r_n\}$ have the same probability distribution. Let r denote the random variable for the remainder length, and denote

$$\pi_i = P\{r = i\}, \qquad i = 0, \ldots, 22 \tag{9.5}$$

and

$$Q_i = P\{\textit{timer\_CU expires} \mid r = i\}, \quad i = 0, \ldots, 22 \tag{9.6}$$

Since an AAL2 cell has 47 byte payload, and for G.723.1 a CPS packet requires 23 bytes, it happens that for all $i = 0, \ldots, 22$

$$Q_i = P\{\textit{less than 2 packets are received in } \tau \textit{ ms}\} \tag{9.7}$$

$$= R_0 + R_1 \tag{9.8}$$

Denote

$$Q = R_0 + R_1 \tag{9.9}$$

then the transition matrix P is

$$\begin{pmatrix} Q & 0 & 0 & 0 & \dots & 0 & 1-Q \\ 1 & 0 & 0 & 0 & \dots & 0 & 0 \\ Q & 1-Q & 0 & 0 & \dots & 0 & 0 \\ Q & 0 & 1-Q & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots \\ Q & 0 & 0 & 0 & \dots & 0 & 0 \\ Q & 0 & 0 & 0 & \dots & 1-Q & 0 \end{pmatrix} \tag{9.10}$$

In the stationary state, we have $\pi = \pi P$, so

$$\pi_0 = Q + \pi_1(1-Q) \tag{9.11}$$

$$\pi_i = \pi_{i+1}(1-Q), \qquad i = 1, 2, \dots, 21 \tag{9.12}$$

$$\pi_{22} = \pi_0(1-Q) \tag{9.13}$$

Therefore,

$$\pi_1 = \pi_0(1-Q)^{22}, \qquad \pi_0 = \pi_0(1-Q)^{23} + Q \tag{9.14}$$

and

$$\pi_0 = \frac{Q}{1-(1-Q)^{23}} \tag{9.15}$$

$$\pi_i = \pi_0(1-Q)^{23-i}, \qquad i = 1, 2, \dots, 22 \tag{9.16}$$

Using these probabilities, the average number of bytes in an AAL2 cell is:

$$C = \pi_0(23R_0 + 46R_1 + 47(1-Q)) \tag{9.17}$$

$$+ \sum_{i=1}^{22} \pi_i(iR_0 + (i+23)R_1 + 47(1-Q)) \tag{9.18}$$

Because each 20-byte voice packet has a 3 byte mini-header and each AAL2 cell has a 6-byte overhead, the packing density is:

$$D = \frac{C}{53} \times \frac{20}{23} \times 100\% \tag{9.19}$$

Notice that

$$\frac{47}{53} \times \frac{20}{23} \times 100\% = 77.11\% \tag{9.20}$$

is the maximum possible density.

9.5 Simulation and Comparison

To validate our analysis, we used a simulation program to actually implement the AAL2 packing process. The simulation model is described in Section 2. The results of 11 simulations are summarized in Table 9.1. The first column is the Timer_CU value used in the simulation. The second column is the total number of cells being sent. Column 3, 4 and 5 respectively are the number of cells that have 0, 1, 2 or more packet arrivals since the packing of the first packet. The actual density is listed in the last column.

τ	total cells	rcvd 0 pkt	rcvd 1 pkt	rcvd 2+ pckt	density (%)
0.5	5621	4079	1246	296	46.93
1	5028	2339	1787	902	56.00
2	4286	949	1462	1875	64.78
3	4077	479	960	2638	69.00
4	3705	154	626	2925	72.43
5	3645	69	315	3261	74.35
6	3587	26	206	3355	75.42
8	3470	8	66	3396	76.54
10	3716	0	10	3706	77.05
12	3779	0	0	3779	77.11
14	3632	0	0	3632	77.11

Table 9.1: Simulation results: number of packets received within τ ms

We calculate the corresponding R_0, R_1, $R_{2^+} = 1 - R_0 - R_1$, and compare them with our analysis results. In Figure 9.4, the analytic probabilities are shown with solid lines and simulation data are shown with "+"s. The resulting packing density for different Timer_CU

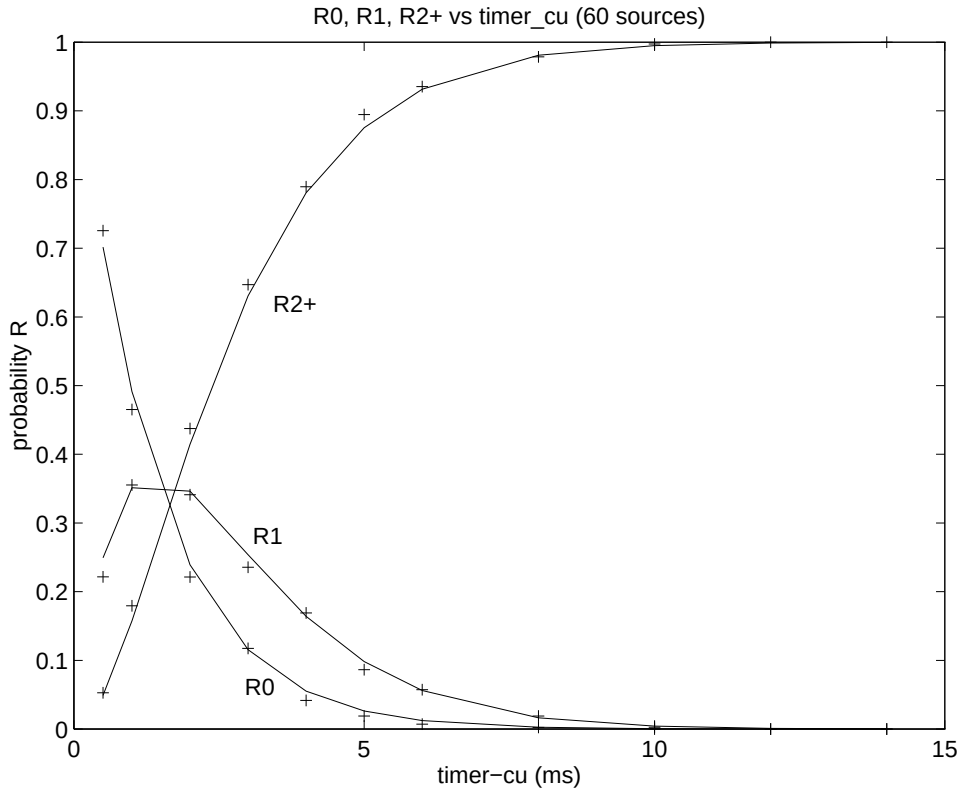


Figure 9.4: R_0, R_1 and R_{2^+} for different Timer_CU values, 60 sources

values is shown in Figure 9.5. Again the analytic calculation is shown with solid line and the simulation data are shown with "+"s. The simulation data match the analytic calculation perfectly.

One application of the analytic formula for density is to find the appropriate Timer_CU value to reach the desired packing density. Figure 9.6 shows the needed Timer_CU values

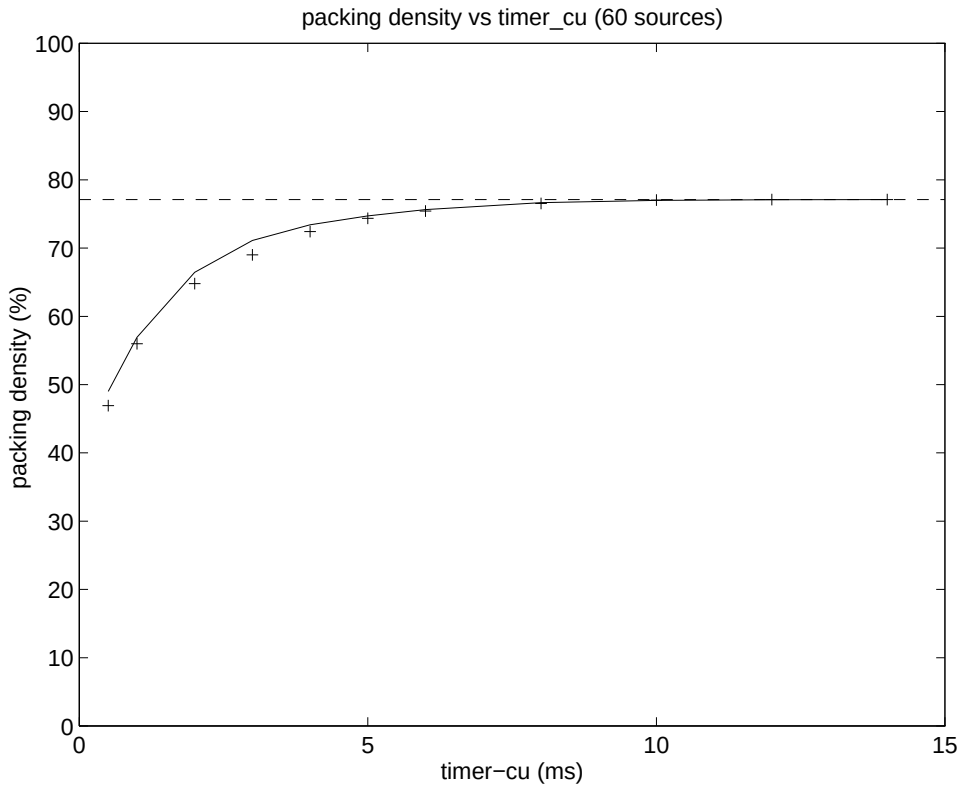


Figure 9.5: Packing density for different Timer_CU values, 60 sources

in order to reach 90% and 95% of the maximum packing density. Timer_CU value also has an impact on the end-to-end delay. Increasing Timer_CU value may cause longer delays. However, this impact is relatively small compared to other components of the total delay. In practice, we should take into account both the desired link efficiency and maximum acceptable delay. The value calculated by the above method can be used as a reference.

9.6 Chapter Summary

In conclusion, we find that Timer_CU value has significant impact on link efficiency. An appropriate choice of Timer_CU value depends on the number of voice sources and

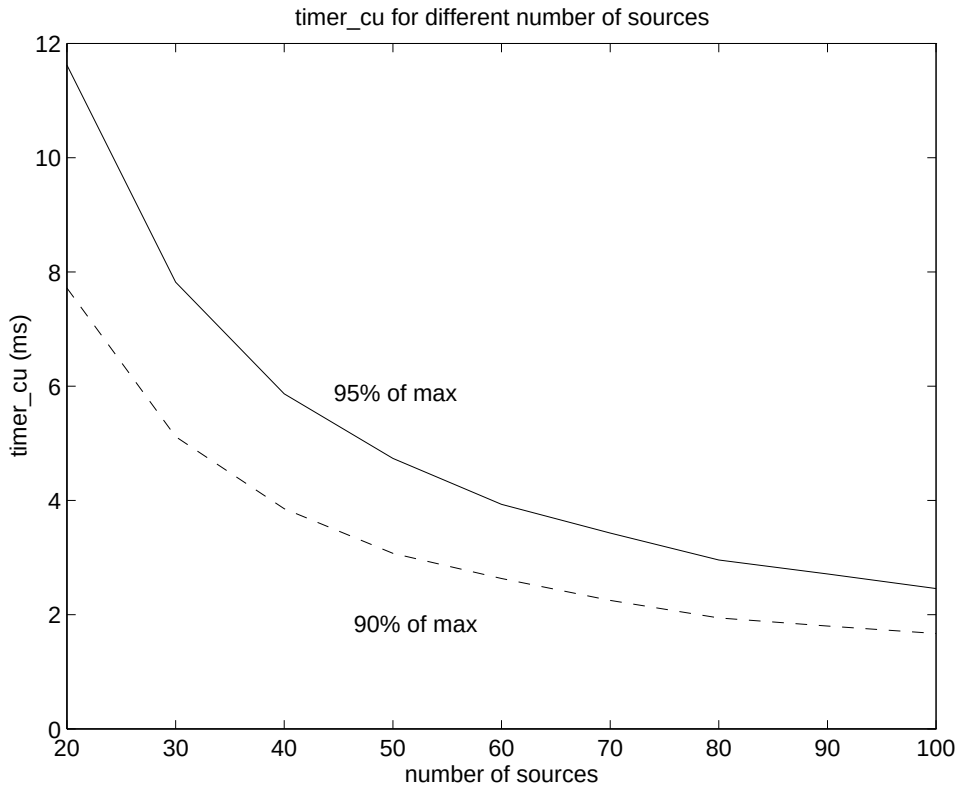


Figure 9.6: Timer_CU values to reach 90% and 95% of the maximum packing density

delay requirements. The analysis of this chapter gives an algorithm to calculate a reference Timer_CU value to achieve the desired link efficiency.

CHAPTER 10

SUMMARY AND OPEN ISSUES

Wireless networks have seen a tremendous growth in the past decade and still keep expanding at a fast pace. While wireless networks provide convenience and mobility in many situations, the performance of wireless networks has been far from satisfactory. This is because the majority of current networking protocols were designed for wired networks. Various assumptions were made based on the characteristics of wired media. When wireless links are added to the networks, the low-bandwidth, shared, error-prone and asymmetric wireless media contradicts these assumptions and causes poor performance.

The goal of wireless enhancements is to identify assumptions and deficiencies in current networking protocols that cause the performance degradation, and to modify these protocols such that wireless networks can achieve high performance and bring high-quality network services to mobile users.

10.1 Key Results

In this study, we obtained the following four major results:

1. **The Congestion Coherence scheme that enhances TCP over wireless links**

The Congestion Coherence scheme we proposed in Chapter 6 is a unique enhancement based on our analysis on the contributors of wireless TCP performance degradation. Using ECN as the congestion feedback mechanism and the "congestion coherence" in consecutive packets, this scheme avoids the majority of end-to-end retransmissions, unnecessary slowdowns and timeouts caused by wireless errors, and therefore improves the performance of TCP over wireless links. From the modification perspective, this scheme only requires minor modification in the TCP code at the mobile station's side. No modification is needed in the base station and in the fixed host, assuming that ECN has been implemented in all network routers. In this way, TCP's end-to-end semantic is maintained; all modifications are in the scope of wireless service providers; the scheme can work with encrypted traffic and applies to two-way traffic. In addition, this scheme applies to intermediate wireless links, such as satellite links and mobile networks.

These key differentiators establish the Congestion Coherence scheme as a unique wireless TCP enhancement. This scheme clearly shows the advantages of ECN over traditional congestion control mechanisms. Since Congestion Coherence highly depends on ECN, we recommend it to be used when ECN is widely deployed. This scheme is a new wireless TCP enhancement as well as a demonstration of the benefits of the ECN protocol. A conclusion that can be drawn from this study is that the ECN standard should be deployed widely to improve network performance.

2. **The mark-front strategy that delivers faster congestion feedback**

The "mark-front strategy" proposed in Chapter 7 is a congestion feedback mechanism that delivers faster congestion feedback than timeout, triple duplicate ACKs

and explicit congestion notification. Faster congestion feedback reduces the number of packet losses during a congestion episode and the delay experienced by packets passing the congested router. It also reduces hardware cost of the routers by requiring smaller buffer sizes. It has been observed that these mechanisms have some bad side effects such as the lock-out phenomenon [24] and unfairness.

3. **A MAC protocol for asymmetric wireless LAN**

   The OSU-MAC, presented in Chapter 8, is a MAC designed for asymmetric forward/reverse channels and half duplex transmission. The key differentiators of this MAC protocol are its ability to sport both real-time and non-real-time applications, and its features to maximize the utilization of the asymmetric bandwidth. The principles and techniques used in this design can apply to high-speed wireless networks and support large-scale real-time applications.

4. **A stochastic analysis to balance voice quality and bandwidth efficiency in AAL2 voice trunking**

   The last major result of this dissertation is the stochastic analysis of AAL2 packing process presented in Chapter 9. The Markovian chain model used to describe the packing process is proved to be a close approximation to the simulation. This model is used to analyze the relationship between voice delay and bandwidth efficiency. Timer_CU value represents the tradeoff between delay and bandwidth efficiency. A guideline for setting the AAL2 Timer_CU value is established and adopted in the 3GPP RAN Technical Specification Group standard: Delay Budget within the Access Stratum [1].

In summary, we identified problems in the protocols used in current wireless networks and proposed enhancements so that these protocols can bring high performance and high quality service to mobile users.

10.2 Open Issues and Future Work

The following are a number of open issues that we will leave for future study.

10.2.1 Implementation in ECN-not-capable regions

The proposed method requires the congested routers in the data path to implement ECN. In reality, this is equivalent to require all routers to implement ECN. Since implementation of ECN in the Internet is gradual, in this project, we need to find alternative ways to implement this scheme when only part of the network is ECN-Capable.

10.2.2 Real Time Congestion Status

The current ECN regards a marked packet as an instance of packet loss. It requires that the sender treat an ECN-Echo essentially the same as a packet loss. The ECN standard reserves a bit in the TCP header to provide robustness against the possibility of a dropped ACK packet carrying an ECN-Echo flag. When a CE packet is received, the receiver sets the ECN-Echo flag in all subsequent ACK packets until a CWR notice is received as shown in Figure 10.1.

We argue that using the CWR bit is neither economical nor justifiable. First, when ECN is deployed, most congestion signals are carried by ECN. Packet drops will be rare. Using the CWR bit to provide robustness against dropped ECN-Echo is not economical. This bit can be put to better use (see next subsection). Second, we think congestion is a real time state. The TCP sender should respond to the most recent feedback from the network instead

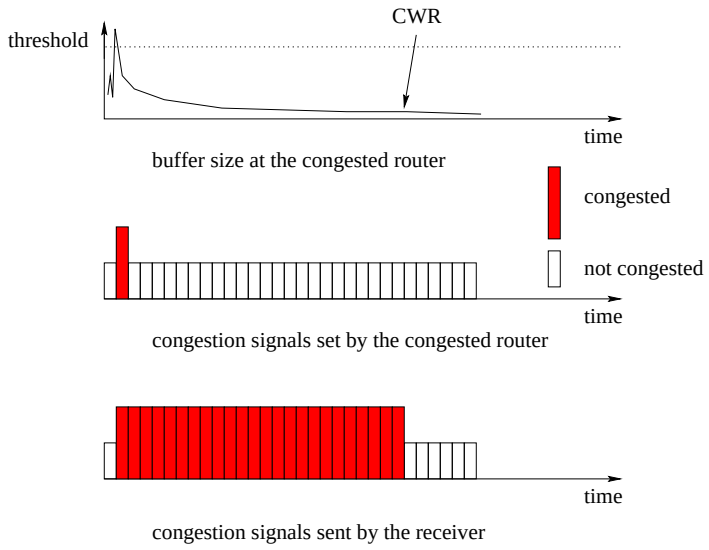


Figure 10.1: Wrong, outdated congestion signals caused by CWR

of the outdated one. If an ECN-Echo is dropped and next ECN signal comes, the sender should respond to the new signal. For example, if a packet is marked as CE but subsequent packets are not marked, the sender should regard this as a passed congestion and not reduce its congestion window.

We propose that the receiver should faithfully reflect the congestion signals set by the intermediate routers. The sender should respond to the latest congestion signal. We will simulate different traffic conditions to see the performance of different policies. We expect that responding to the latest congestion signal can make the best use of link capacity, maintain low transmission delay, and avoid most packet losses.

10.2.3 Three-Level ECN

Current ECN uses two bits in the IP header [66]. The first bit is called ECT (ECN-Capable Transport) bit. It is set if both sender and receiver are ECN capable and wish to

use ECN. The second bit is called the CE (Congestion Experienced) bit. If the ECT bit is set in a packet, the router can set the CE bit to indicate congestion. This scheme is not efficient, because it uses two bits indicate three states: not ECT; ECT and congested; and ECT and not congested. Since two bits can represent four states, a more efficient scheme is possible.

Based on the above observation and the proposal in previous subsection, we propose to change the ECN field in the IP header and ECN-Echo field in the TCP header. The ECN field consists of two bits and the ECN-Echo field consists of two bits. The ECN field is set by the source and changed by the intermediate routers, the ECN-Echo field is set by the receiver. The proposed bit patterns are summarized in Table 10.2.3. With these bit patterns, network routers can indicate three levels of congestion with ECN signals: no congestion, mild congestion and severe congestion. The receiver can then faithfully reflect the ECN signals in the ECN-Echo field. Use of this feedback is left to the sender TCP.

ECN	**ECN field meaning**
00	not ECN capable
01	ECN capable and no congestion
10	ECN capable and mild congestion
11	ECN capable and severe congestion

ECN-Echo	**ECN-Echo field meaning**
00	reserved for other use
01	echo of no congestion
10	echo of mild congestion
11	echo of severe congestion

Table 10.1: Proposed ECN and ECN-Echo fields

The routers set congestion signals based on two thresholds, one for mild congestion and another for severe congestion. A router marks the packet according to its current congestion status, but if the packet already carries a higher level mark, it will not change the mark.

Since more informational congestion signals are provided, the sender responds to the ECN signals according to the following rules:

- If the signal is "no congestion", the sender follows existing slow start or congestion avoidance algorithms to increase the window size.
- If the signal is "mild congestion" and there was no window reduction in the past RTT, the sender decreases its congestion window to αCWND, where CWND is the current congestion window.
- If the signal is "severe congestion" and there was no window reduction in the past RTT, the sender decreases its congestion window to βCWND.
- If the signal is "severe congestion" and there was a window reduction caused by "mild congestion" but no window reduction due to "severe congestion" in the past RTT, then the sender further reduces the congestion window to β/α CWND.

Here, $0 < \beta < \alpha < 1$ are two constants, their values are subject to further study.

The principle in the window reduction is that the sender should reduce its congestion window at most once in an RTT, but a "severe congestion" signal can override a previous "mild congestion" signal.

The above algorithm needs further validation and verification. Since more information is conveyed from the network, we expect the new scheme will have higher throughput and lower buffer occupancy.

BIBLIOGRAPHY

[1] 3GPP. Delay budget within the access stratum, technical specification group ran, 3gpp tr 25.853 v3.1.0, 2000.

[2] M. Allman, V. Paxson, and W. Stevens. TCP congestion control. *RFC 2581*, Apr. 1999.

[3] N. Amitay. Distributed Switching and Control with Fast Resource Assignment/Handoff for Personal Communications Systems. *IEEE JSAC*, 11:842–849, 1993.

[4] ATM Forum. ATM trunking using aal2 for narrowband services,af-vtoa-0113.000, 1999.

[5] A. Bakre and B. R. Badrinath. I-TCP: Indirect TCP for mobile hosts. *15th International Conference on Distributed Computing Systems*, 1995.

[6] H. Balakrishnan and R. Katz. Explicit loss notification and wireless web performance. *Proc. IEEE Globecom Internet MiniConference, Sydney, Australia, November 1998.*, 1998.

[7] S. Banerjee and J. Goteti. Extending tcp for wireless networks. *University of Maryland, College Park*, 1997.

[8] S. Biaz, M. Mehta, S. West, and N. H. Vaidya. TCP over wireless networks using multiple acknowledgements. *Technical Report 97-001, Computer Science, Texas A&M University*, 1997.

[9] S. Biaz and N. H. Vaidya. Discriminating congestion losses from wireless losses using inter-arrival times at the receiver. *IEEE Symposium ASSET'99, Richardson, TX, USA*, March 1999.

[10] R. E. Blahut. *Theory and practice of error control codes.* Addison-Wesley, Reading, MA, 1983.

[11] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An architecture for diferentiated services, 1998.

[12] R. T. Braden. RFC 1122: Requirements for Internet hosts — communication layers, Oct. 1989.

[13] P. T. Brady. A model for generating on-off speech patterns in two-way conversation. *Bell System Technical Journal*, pages 2445–2472, September 1969.

[14] L. S. Brakmo, S. W. O'Malley, and L. L. Peterson. TCP Vegas: New techniques for congestion detection and avoidance. In *Proceedings, 1994 SIGCOMM Conference*, pages 24–35, London, UK, Aug. 31st - Sept. 2nd 1994.

[15] CableLabs. Data Over Cable Service Interface Specification. http://www.cablemodem.com/public/pubtech.html, April 1998.

[16] C. Chen, H. Krishnan, S. Leung, and N. Tang. Implementing explicit congestion notification (ecn) in tcp for ipv6, Dec. 1997.

[17] K. Cheon and S. Panwar. The performance of atm-ubr with early selective packet discard, 1998.

[18] D. Chiu and R. Jain. Analysis of the increase and decrease algorithms for congestion avoidance in computer networks. *Computer Networks and ISDN Systems*, 17:14, 1989.

[19] C.L.Fullmer and J.Garcia-Luna-Aceves. Floor Acquisition Multiple Access (FAMA) for Packet-Radio Networks. In *Proceedings of ACM SIGCOMM '95*, 1995.

[20] D. Cohen. Issues in transnet packetized voice communications, 1977.

[21] D. Cohen, f the, and V. Protocol. Arpanet working group requests for comment, 1976.

[22] S. Deng. Traffic characteristics of packet voice. *IEEE International Conference on Communications*, 3:1369–1374, 1995.

[23] C. DWRED. Distributed weighted random early detection.

[24] B. B. et al. Recommendations on queue management and congestion avoidance in the internet. *RFC 2309*, page 16, 1998.

[25] N. R. Figueira and J. Pasquale. Remote-Queueing Multiple Access (RQMA): Providing Quality of Service for Wireless Communications. In *Proc. IEEE INFOCOM'98*. IEEE Computer Society, April 1998.

[26] S. Floyd. TCP and explicit congestion notification. *ACM Computer Communication Review*, 24(5):10–23, 1994.

[27] S. Floyd. The NewReno Modification to TCP's Fast Recovery Algorithm. *RFC 2582*, Apr. 1999.

[28] S. Floyd. Red with drop from front, email discussion on the end2end mailing list, March 1998.

[29] S. Floyd. The gentle option in ns, March 2000.

[30] S. Floyd and V. Jacobson. Traffic phase effects in packet-switched gateways. *ACM SIGCOMM comp. commun. rev.*, 21, 2:26–42, 1991.

[31] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, 1(4):397–413, Aug. 1993.

[32] R. Gibbens and F. Kelly. Resource pricing and congestion control, 1999.

[33] R. Goyal, R. Jain, S. Kalyanaraman, S. Fahmy, and S.-C. Kim. UBR+: Improving performance of TCP over ATM-UBR service. In *First IEEE Enterprise Networking Mini-Conference (ENM-97) in conjunction with the ICC-97*, pages 1042–1048, 1997.

[34] S. S. H. Balakrishnan and R. H. Katz. Improving reliable transport and handoff performance in cellular wireless networks. *Wireless Networks*, 1(4):469 – 481, February 1995.

[35] G. H. Heilmeier. Global begins at home. *IEEE Communications Magazine*, 30(10):50–56, October 1992.

[36] J. C. Hoe. Improving the start-up behavior of a congestion control sheme for TCP. In *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, volume 26,4 of *ACM SIGCOMM Computer Communication Review*, pages 270–280, New York, Aug. 26–30 1996. ACM Press.

[37] ITU. General recommendation on the transmission quality for an entire international telephone connection; one-way transmission time, recommendation g.114, march 1993., 1993.

[38] ITU. AAL type 2 service specific convergence sublayer for trunking, recommendation i.366.2, feb. 1999, 1999.

[39] V. Jacobson. Congestion avoidance and control. *ACM Computer Communication Review; Proceedings of the Sigcomm '88 Symposium in Stanford, CA, August, 1988*, 18, 4:314–329, 1988.

[40] V. Jacobson, R. Braden, and D. Borman. Rfc 1323: Tcp extensions for high performance, 1992.

[41] R. Jain. A timeout-based congestion control scheme for window flow-controlled networks. *IEEE Journal on Selected Areas in Communications*, SAC-4, 7:1162–1167, 1986.

[42] R. Jain. *The Art of Computer Systems Performance Analysis*. John Wiley & Sons, Inc., New York, NY, 1991.

[43] R. Jain. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling,*. Wiley-Interscience, New York, NY, May 1991.

[44] P. Karn and C. Partridge. Improving Round-Trip Time Estimates in Reliable Transport Protocols. In *Proceedings, SIGCOMM '87 Workshop*, pages 2–7. ACM SIGCOMM, ACM Press, Aug. 1987. Stowe, Vermont.

[45] P. Karn and C. Partridge. Improving round-trip time estimates in reliable transport protocols. *ACM Transactions on Computer Systems*, 9(4):364–373, Nov. 1991.

[46] Y. Kim and S. qi Li. Performance analysis of data packet discarding in atm networks, 1997.

[47] M. Kojo, K. E. E. Raatikainen, M. Liljeberg, J. Kiiskinen, and T. O. Alanko. An efficient transport service for slow wireless telephone links. *IEEE Journal of Selected Areas in Communications*, 15(7):1337–1348, 1997.

[48] M. Labrador and S. Banerjee. Enhancing application throughput by selective packet dropping, 1999.

[49] T. Lakshman and U. Madhow. Performance analysis of window-base flow control using tcp/ip: The effect of high bandwidth-delay products and random loss, 1994.

[50] T. Lakshman, A. Neidhardt, and T. Ott. The drop from front strategy in tcp over atm and its interworking with other control features.

[51] T. Lakshman, A. Neidhardt, and T. Ott. The drop from front strategy in tcp and in tcp over atm, 1996.

[52] Y. Lapid, R. Rom, and M. Sidi. Analysis of packet discarding policies in high speed networks, 1997.

[53] LBNL. Network simulator ns, 2001.

[54] D. Lin and H. T. Kung. TCP fast recovery strategies: Analysis and improvements. In *Proc. INFOCOM 98*, 1998.

[55] C. Liu and R. Jain. Improving explicit congestion notification with the mark-front strategy. *Computer Networks*, 35(2–3):185–201, February 2001.

[56] D. Magill. Adaptive speech compression for packet communication systems, 1973.

[57] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow. RFC 2018: TCP selective acknowledgment options, Oct. 1996. Status: PROPOSED STANDARD.

[58] A. J. McAuley. Reliable broadband communication using a burst erasure correcting code. In *Proc. of ACM SIGCOMM*, pages 297–306, September 1990.

[59] S. Nanda, D. Goodman, and U. Timor. Performance of PRMA: A Packet Voice Protocol for Cellular Systems. *IEEE Trans. Veh. Techn.*, 40:584–598, 1991.

[60] J. Pan, J. W. Mark, and X. Shen. Tcp performance and its improvement over wireless links, 2000.

[61] V. Paxson and M. Allman. RFC 2988: Computing TCP's Retransmission Timer. *IETF*, Nov. 2000.

[62] J. Postel. Internet control message protocol icmp, 1981.

[63] J. Postel. RFC 793: Transmission control protocol, Sept. 1981.

[64] X. Qiu and V. O. Li. Dynamic Reservation Multiple Access (DRMA): A New Multiple Access Scheme for Personal Communication System (PCS). *Wireless Networks*, 2:117–128, 1996.

[65] K. Ramakrishnan and S. Floyd. A proposal to add explicit congestion notification, 1998.

[66] K. Ramakrishnan and S. Floyd. RFC 2481: A proposal to add Explicit Congestion Notification (ECN) to IP, Jan. 1999.

[67] K. K. Ramakrishnan and R. Jain. A binary feedback scheme for congestion avoidance in computer networks with a connectionless network layer. *Proceedings of the 1988 SIGCOMM Symposium on Communications Architectures and Protocols; ACM; Stanford, CA*, pages 303–313, 1988.

[68] A. Romanow and S. Floyd. The dynamics of TCP traffic over ATM networks. In *Proceedings, 1994 SIGCOMM Conference*, pages 79–88, London, UK, 31st - 2nd 1994.

[69] J. Salim and U. Ahmed. Performance evaluation of explicit congestion notification, 2000.

[70] W. Stevens. RFC 2001: TCP slow start, congestion avoidance, fast retransmit, and fast recovery algorithms, Jan. 1997. Status: PROPOSED STANDARD.

[71] W. Stevens. Tcp slow start, congestion avoidance, fast retransmit, and fast recovery algorithms, 2001.

[72] H.-Y. Tyan, B. Wang, Y. Ye, L. Su, W. Lin, and C.-J. Hou. NetSim^Q: a java-integrated network simulation tool for qos control in high speed networks. http://eewww.eng.ohio-state.edu/drcl/grants/middleware97/netsimQ.html., 1999.

[73] N. Vaidya, M. Mehta, C. Perkins, and G. Montenegro. Delayed duplicate acknowledgements: A TCP-unaware approach to improve performance of TCP over wireless. *Technical Report, Computer Science Dept., Texas A&M University*, 1999.

[74] N. Wilson, R. Ganesh, K. Joseph, and D. Raychaudhuri. Packet CDMA Versus Dynamic TDMA for Multiple Access in An Integrated Voice/Data PCN. *IEEE JSAC*, 11:870–884, 1993.

[75] G. Xylomenos. Multi service link layers: An approach to enhancing internet performance over wireless links, 1999.

[76] N. Yin and M. G. Hluchyj. Implication of dropping packets from the front of a queue. *Proc. 7th ITC seminar*, page 10.4, 1990.

[77] K. Zhang. Packet delay variation in voice trunking using aal2, 1998.

[78] S. S. L. Zhang and D. Clark. Observations on the dynamics of a congestion control algorithm: The effects of two-way traffic. *ACM SIGCOMM 91*, 1991.

ACRONYMS

AAL	ATM Adaptation Layer
ABR	Available Bit Rate
ACK	Acknowledgment
ANOVA	Analysis Of Variations
ATM	Asynchronous Transfer Mode
BECN	Backward Explicit Congestion Notification
B-ISDN	Broadband Integrated Services Digital Network
BNR1	BSD Network Release 1
BNR2	BSD Network Release 2
BS	Base Station
BSD	Berkley Software Distribution
CAC	Connection Admission Control
CBR	Constant Bit Rate
CC	Congestion Coherence
CID	Channel ID
CLP	Cell Loss Priority
CLR	Cell Loss Ratio
CSMA	Carrier Sense Multiple Access
CWND	TCP Congestion Window
CWR	Congestion Window Reduced
DDA	Delayed Duplicate Acknowledgment
DOCSIS	Data Over Cable Service Interface Specification
DRMA	Dynamic Reservation Multiple Access
DTDMA	Dynamic TDMA
ECN	Explicit Congestion Notification
ECT	ECN Capable Transport
EFCI	Explicit Forward Congestion Indication
EIN	Equipment Identification Number
ELN	Explicit Loss Notification
EOM	End Of Message
EPD	Early Packet Discard
ER	Explicit Rate
FAMA	Floor Acquisition Multiple Access
FEC	Forward Error Correction

FH	Fixed Host
FIFO	First In First Out
FTP	File Transfer Protocol
GF	Galios Field
GPS	Global Positioning System
GSM	Global System for Mobile Communication
HEC	Header Error Control
ICMP	Internet Control Message Protocol
IETF	Internet Engineering Task Force
IP	Internet Protocol
I-TCP	Indirect TCP
ITU	International Telecommunications Union
LAN	Local Area Network
LI	Length Indicator
MAC	Media Access Control
MCNS	Multimedia Cable Network System
MH	Mobile Host
NAK	Negative acknowledgment
PCS	Personal Communications Service
PDFD	Partial Frame Drop at the Front
PPD	Partial Packet Discard
PRMA	Packet Reservation Multiple Access
PSK	Phase Shifting Keying
PS	Pilot Symbol
QoS	Quality of Service
QPSK	Quaternary PSK Modulation
RAMA	Resource Auction Multiple Access
RED	Random Early Detection
RFC	Request For Comments
RQMA	Remote-Queuing Multiple Access
RS	Reed-Solomon
RSVP	Reservation Protocol
RTO	Retransmission Timeout
RTT	Round-Trip Time
SIF	Segment In Flight
SSTHRESH	Slow Start Threshold
STF	Start Field
TCP	Transmission Control Protocol
TDMA	Time Division Multiple Access
UDP	User Datagram Protocol
UTP	Unshielded Twisted Pair
UUI	User-to-User Indication
WAN	Wide Area Network

WRED	Weighted Random Early Detection
WWW	World Wide Web

Wireless Network Enhancements Using Congestion Coherence, Faster Congestion Feedback, Media Access Control and AAL2 Voice Trunking

By

Chunlei Liu, Ph.D.

The Ohio State University, 2001

Professor Raj Jain, Adviser

Rapid development of wireless communications in recent years has imposed great challenges for network support. As most current networking protocols were designed mainly for wired networks, many assumptions that make these protocols efficient in wired networks are no longer true and cause severe performance degradation in wireless environment. An urgent task for today's networking research is to identify such deficiencies and to enhance these protocols.

This dissertation contains four major enhancement results. The first result is a wireless TCP enhancement scheme called *Congestion Coherence*. Through a statistical analysis, we find the leading contributor of wireless TCP performance degradation is the timeout resulting from frequent packet losses. This scheme uses ECN to reduce number of timeouts, and based on the observation that congestion neither happens nor disappears suddenly,

uses the sequential coherence of packet marking to determine cause of packet losses. Simulations show this scheme works better than existing enhancements and produces good performance.

The second result is a *Mark-Front Strategy* to deliver faster congestion feedback in networks that use the newly standardized ECN protocol. By choosing to mark the packet in the front of the queue, Mark-Front Strategy delivers faster feedback than traditional congestion feedback mechanisms. Our analysis show Mark-Front Strategy requires smaller buffers, and yields higher throughput and better fairness.

A MAC protocol to supports real-time applications in an asymmetric and half-duplex environment is our third result. Several innovative features are used to ensure optimal utilization of the asymmetric bandwidth.

In the last result, we use a Markovian chain to model the AAL2 voice trunking process and analyze the tradeoff between delay and bandwidth efficiency. This analysis gives a guideline for setting the packing timer. The result is adopted by a 3GPP industrial standard.

This study provides insight into congestion control and quality of service in wireless networks. The proposed enhancements will help to bring high-performance and high-quality services to mobile subscribers.

Wireless Network Enhancements Using Congestion Coherence, Faster Congestion Feedback, Media Access Control and AAL2 Voice Trunking

By

Chunlei Liu, Ph.D.

The Ohio State University, 2001

Professor Raj Jain, Adviser

Rapid development of wireless communications in recent years has imposed great challenges for network support. As most current networking protocols were designed mainly for wired networks, many assumptions that make these protocols efficient in wired networks are no longer true and cause severe performance degradation in wireless environment. An urgent task for today's networking research is to identify such deficiencies and to enhance these protocols.

This dissertation contains four major enhancement results. The first result is a wireless TCP enhancement scheme called *Congestion Coherence*. Through a statistical analysis, we find the leading contributor of wireless TCP performance degradation is the timeout resulting from frequent packet losses. This scheme uses ECN to reduce number of timeouts, and based on the observation that congestion neither happens nor disappears suddenly,

uses the sequential coherence of packet marking to determine cause of packet losses. Simulations show this scheme works better than existing enhancements and produces good performance.

The second result is a *Mark-Front Strategy* to deliver faster congestion feedback in networks that use the newly standardized ECN protocol. By choosing to mark the packet in the front of the queue, Mark-Front Strategy delivers faster feedback than traditional congestion feedback mechanisms. Our analysis show Mark-Front Strategy requires smaller buffers, and yields higher throughput and better fairness.

A MAC protocol to supports real-time applications in an asymmetric and half-duplex environment is our third result. Several innovative features are used to ensure optimal utilization of the asymmetric bandwidth.

In the last result, we use a Markovian chain to model the AAL2 voice trunking process and analyze the tradeoff between delay and bandwidth efficiency. This analysis gives a guideline for setting the packing timer. The result is adopted by a 3GPP industrial standard.

This study provides insight into congestion control and quality of service in wireless networks. The proposed enhancements will help to bring high-performance and high-quality services to mobile subscribers.