# Cross-Layer Protocol Interactions in Heterogeneous Data Networks

by

Chunmei Liu

Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in the field of Computer Systems

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2005



Author .....
Department of Aeronautics and Astronautics
December 08, 2004

Certified by .....
Eytan H. Modiano
Associate Professor of Department of Aeronautics and Astronautics
Thesis Supervisor

Certified by .....
Vincent W. S. Chan
Professor of Department of Electrical Engineering and Computer Science,
and Aeronautics and Astronautics
Director, Laboratory for Information and Decision Systems

Certified by .....
Charles Rohrs
Research Scientist, Department of Electrical Engineering and Computer Science

Accepted by .....
Jaime Peraire
Professor of Aeronautics and Astronautics
Chair, Committee on Graduate Students

# Cross-Layer Protocol Interactions in Heterogeneous Data Networks

by

Chunmei Liu

Submitted to the Department of Aeronautics and Astronautics
on December 08, 2004, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in the field of Computer Systems

## Abstract

Modern data networks are heterogeneous in that they often employ a variety of link technologies, such as wireline, optical, satellite and wireless links. As a result, internet protocols, such as Transmission Control Protocol (TCP), that were designed for wireline networks, perform poorly when used over heterogeneous networks. This is particularly the case for satellite and wireless networks which are often characterized by high bandwidth-delay product and high link loss probability. This thesis examines the performance of TCP in the context of heterogeneous networks, particularly focusing on interactions between protocols across different layers of the protocol stack.

First, we provide an analytical framework to study the interaction between TCP and link layer retransmission protocols (ARQ). The system is modelled as a Markov chain with reward functions, and detailed queueing models are developed for the link layer ARQ. The analysis shows that in most cases implementing ARQ can achieve significant improvement in system throughput. Moreover, by proper choice of protocols parameters, such as the packet size and the number of transmission attempts per packet, significant performance improvement can be obtained.

We then investigate the interaction between TCP at the transport layer and ALOHA at the MAC layer. Two equations are derived to express the system performance in terms of various system and protocol parameters, which show that the maximum possible system throughput is $1/e$. A sufficient and necessary condition to achieve this throughput is also presented, and the optimal MAC layer transmission probability at which the system achieves its highest throughput is given. Furthermore, the impact of other system and protocol parameters, such as TCP timeout backoff and MAC layer retransmissions, are studied in detail. The results show that the system performance is a balance of idle slots and collisions at the MAC layer, and a tradeoff between packet loss probability and round trip time at the transport layer.

Finally, we consider the optimal scheduling problem with window service constraints. Optimal policies that minimize the average response time of jobs are derived and the results show that both the job lengths and the window sizes are essential to the optimal policy.

Thesis Supervisor: Eytan H. Modiano
Title: Associate Professor of Department of Aeronautics and Astronautics

## Acknowledgments

I would like to gratefully express my thanks to all people who have had influence on my research here at MIT. First and foremost, I would like to thank my advisor, Prof. Eytan Modiano, for his invaluable guidance and advice throughout the course of this research, as well as his continuous encouragement and support.

I also would like to thank all LIDS professors and students. It is their lectures and discussions that inspired most of my ideas on proposing and solving the research problems. They also helped to make my stay here a pleasant one.

I would like to acknowledge that this research has been supported and funded by NASA Space Communication Project with grant number NAG3-2835. Without this support, it would be impossible for me to finish the work.

Last but not least, I would like to thank my husband, parents and sisters, who are forever my sources of inspiration, encouragement, and support.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

## 1.1 Background

### 1.1.1 History of Data Networks

Early forms of data networks date back to the smoke signals used by primitive societies [12]. A more recent but still ancient example is the beacon towers along the Great Wall in China more than two thousands years ago. There upon enemy approaching, the soldiers at the frontier lightened their beacon fire. With significant fast speed, this signal was relayed and spread by lighting of other beacon fires along the Great Wall, and eventually reached everywhere throughout the region around the Great Wall. There the fire corresponds to the binary signals in today's data networks, the beacon towers correspond to nodes, and the routing protocol is flooding.

A major development of modern data networks was the development of ARPANET around 1970, which is the first large-scale, general-purpose data network that connected a number of geographically distributed computers together. The technologies ARPANET implemented includes packet switching [31, 11, 32], layered protocols, mesh network topology and across-the-network flow control [23, 24], all of which contribute significantly to today's data networks. Subsequently, many other networks were developed, such as TYMNET and DECNET. Due to the technologies available at that time, these early networks were all wireline networks. Yet many of their ideas and technologies, such as the layered architecture and the TCP/IP (Transmission Control Protocol/Internet Protocol) protocol suite, are still embodied in today's networks that have a much more heterogenous nature,

where heterogeneity refers to the fact that the networks have multiple types of links, such as wireline links, wireless links and satellite links.

Each of these early networks was designed and manufactured by different groups or companies and had different architecture and protocols. Consequently, computers within one network could only communicate with other computers within the same network. In order to allow interconnection of networks from different manufacturers, in late 1970s, the International Standards Organization (ISO) developed the Open Systems Interconnection (OSI) model [62, 23], which was soon widely accepted as an international standard for data networks.

The OSI model has a layered architecture [62]. The basic idea of the layered architecture is to decompose functions at each node into layers. Each layer treats other layers as black-boxes with certain inputs, outputs and their functional relation, and talks to the corresponding layer of other nodes. In particular, the OSI model has seven layers, as illustrated in Figure 1-1. From bottom to top, the seven layers are physical layer, data link control layer (DLC), network layer, transport layer, session layer, presentation layer and application layer, respectively. The DLC has one sublayer called Medium Access Control (MAC) layer that manages multi-access links.

Each layer in the OSI model performs a given function in support of the overall function of the system. Specifically, the physical layer is the lowest layer and is responsible for the actual transmissions of bits over a physical link. It is essentially a bit pipe. The second layer, the data link control layer, is responsible for packet transmissions across individual links. Its sub-layer, the MAC layer, allocates multi-access channels so that nodes can transmit with minimal interference from other nodes. The third layer, the network layer, performs routing and some congestion control and provides a virtual link for end-to-end packets. The main function of the fourth layer, the transport layer, is to provide end-to-end congestion control and reliable end-to-end message transmissions upon session's requirement. The upper three layers, the session, the presentation, and the application layer, are beyond the scope of this thesis. We view them as layers that send data down to the transport layer.

On one hand, the layered architecture has many advantage, such as simplicity of design, understandability, and standard, interchangeable, widely available modules. It allows easy interoperation among different networks as well. On the other hand, layered architecture also introduces inefficiencies in the form of cross-layer protocol interactions. Since many

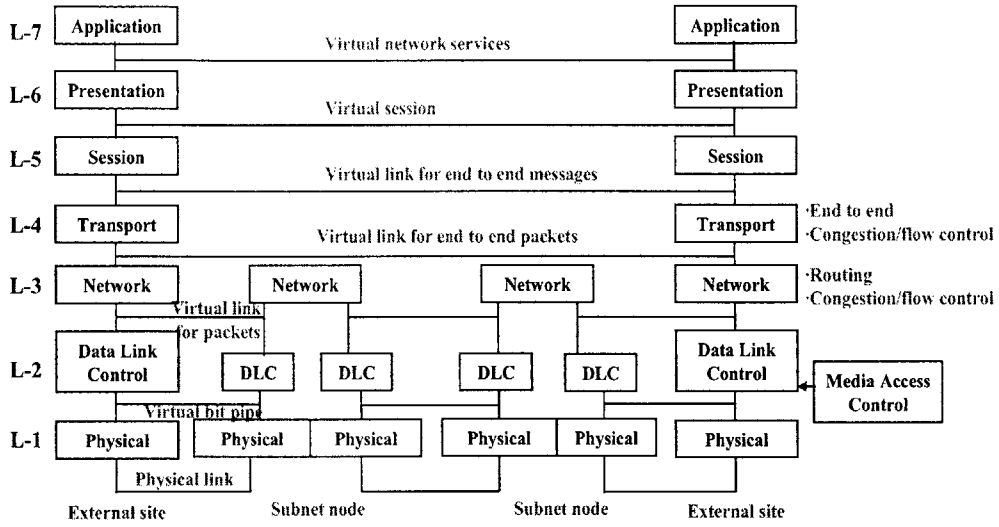


Figure 1-1: OSI Model - Layered Architecture

protocols originated from wireline networks, such as the popular TCP/IP suite, these cross-layer protocol interactions have been taken into account for wireline networks during the design phase, and wireline networks hence perform reasonably well under these protocols. However, with the development of new link technologies, such as satellite links and wireless links, current networks have a more heterogenous nature. For these heterogenous networks, there is a growing body of research indicating that these cross-layer protocol interactions may significantly hurt system performance [4, 50, 49, 45, 7, 27, 3, 10]. The focus of this thesis is to analyze these cross-layer protocol interactions in heterogenous data networks.

### 1.1.2 Heterogenous Data Networks

In the 1950s, when people started to connect computers in a centralized way, the link connecting different terminals were telephone links (300-1200bps) [12]. Nowadays the most common media include twisted pair (1.544Mbps), coaxial cable (multi-Mbps), optical fiber(Gbps), radio (multi-kbps to 1Mbps), satellite (multi-kbps to Gbps), and microwave (multi-Gbps). In the past two decades, great efforts have been made towards integrating different links into one network. As a result, current data networks have a much more heterogenous nature, that is, they include various types of links, such as the wireline links, wireless links, satellite links or optical links. In this thesis, we consider heterogenous net-

works that have wireline links, satellite links and/or wireless links.

Compared to traditional wireline networks, the heterogenous networks considered have many special features, two of which are high bandwidth-delay product due to the large propagation delay of satellite links and high link loss probability which is typical of both wireless and satellite links. The cross-layer protocol interactions due to these features may affect system performance significantly [4, 50, 49, 45, 7, 27, 3, 10]. In this thesis we focus on two forms of cross-layer protocol interactions: the interaction between transport layer and data link layer, and the interaction between transport layer and MAC layer.

Currently the dominate transport layer protocol is Transmission Control Protocol (TCP), and the dominate reliable data link layer protocol uses Automatic Repeat reQuest (ARQ). They have duplicate functions, that is, both TCP and ARQ perform loss recovery. In addition, the retransmissions of packets by ARQ at the data link layer introduces variability of Round Trip Time (RTT) seen by the transport layer packets. This variability can be very large in networks with high bandwidth-delay product, such as hybrid terrestrial-satellite networks, and could potentially cause TCP timeouts that significantly degrade the performance of TCP. These TCP timeouts were designed to relieve network congestion. The adverse effect of initiating a TCP timeout due to link layer retransmissions is just one example of the kind of interactions that we will study in this thesis. In particular, in Chapter 3, we will investigate in detail the interactions between TCP and various link layer protocols.

Another form of interactions considered is the interaction between the transport layer and the MAC layer. When a random access protocol such as ALOHA is employed at the MAC layer, collisions occur with high probability. These MAC collisions directly affects the TCP window evolution at the transport layer. Moreover, since the MAC layer accepts packets from its corresponding higher layers, TCP window flow control algorithm controls the packet arrival process at the MAC layer. Chapter 4 will discuss these interactions between TCP and random access protocols in detail.

There are other cross-layer interactions in heterogenous networks, such as that among network layer, data link layer and physical layer due to the time-varying network topology and channel conditions in satellite networks and wireless networks. In this thesis we will focus on the interactions between TCP and ARQ as well as TCP and ALOHA. Furthermore, we also examines the problem of optimal transmission scheduling when packets availability is limited by window constraints, for example, due to TCP window size limitation. This

subject will be addressed in Chapter 5.

## 1.2 Thesis Contributions

In this thesis we examine the cross-layer protocol interactions between transport layer and data link layer as well as transport layer and MAC layer. The thesis consists of three main parts: the interaction between TCP and ARQ, the interaction between TCP and ALOHA, and optimal scheduling policy with window service constraints.

### 1.2.1 Interaction between TCP and ARQ

The topic of interaction between TCP and ARQ comes directly from the observation of poor TCP performance in heterogenous data networks with high link loss probability and high bandwidth-delay product, such as hybrid satellite-terrestrial networks. We will focus on this type of networks in exploring the interaction between TCP and ARQ. Our discussions on high link loss probability is also applicable for heterogenous data networks with high link loss probability but no high bandwidth-delay product, such as wireless networks. For readers not familiar with TCP and ARQ, a brief introduction will be given in Chapter 2.

Currently TCP is the dominate reliable transport layer protocol [52]. On the one hand, TCP works very well in classical wireline networks, as evident by the current Internet. On the other hand, TCP does not perform well in heterogenous data networks, especially those with high link loss probability and high bandwidth-delay product [4, 50, 49, 45]. There are two major reasons for TCP's poor performance in heterogenous data networks. One is that TCP considers the link losses in such networks as congestion losses and consequently reduces window size. Another reason is that the window increase rate in TCP congestion avoidance phase is slow (ref. Chapter 2 Section 2.3). The high bandwidth-delay product property means that this slow increasing rate takes TCP long time to raise its window size comparable to the available network capacity. Combining these two factors, it is foreseeable that in such networks, with high probability, TCP will operate at a window size far below the available network capacity. As a result, the overall TCP performance is significantly deteriorated.

There are mainly three solutions proposed to enhance TCP performance over this type of heterogenous networks: TCP spoofing [8], split-TCP [6, 59], and link layer solutions.

Although their detail operations are different, their main ideas are the same: solving the link losses locally instead of leaving them to TCP. Comprehensive descriptions can be found in [7, 27, 45, 3, 10].

Among all these solutions, link layer solution has the advantage that it fits naturally into the layered structure of networks. The main idea is that it can "hide" the link losses from TCP, hence greatly reduce the number of unnecessary TCP window reductions and significantly improve the overall TCP performance. Two well-known link-layer solutions are Forward-Error-Correction (FEC) and Automatic Repeat reQuest (ARQ). Since FEC has the advantage that no retransmissions are needed to recover the link losses, it is a good solution when it can be implemented efficiently. However, when the channel condition is time-varying as in some heterogenous data networks with wireless links or satellite links, it is usually hard to design and implement FEC code efficiently. As a result, the packet loss probability could be significant even after FEC. In these scenarios ARQ becomes a necessary option.

ARQ recovers link losses by retransmitting lost packets (See Chapter 2 Section 2.1 for detail operations of ARQ). By doing so, ARQ can "hide" the link losses from TCP, hence significantly improving the overall TCP performance. On the other hand, the presence of an ARQ protocol can lead to other more subtle problems. In particular, due to the high bandwidth-delay product, ARQ retransmissions introduce high variability of the packet round trip time (RTT) seen by TCP, and lead to TCP timeouts due to ARQ retransmissions. We call these timeouts false timeouts.

Recently a number of papers have examined the interaction between TCP and link layer ARQ by examining the system performance with TCP at the transport layer and ARQ at the link layer. Most of them involve simulations in wireless environment [7, 19, 20, 39, 58, 5, 13]. There is some analytical work on this topic as well [14, 16, 17, 18, 43]. These papers provide approximate analysis of TCP performance over a link layer protocol with the assumption of instantaneous ACK feedback. The large bandwidth-delay product of heterogenous networks we consider here makes these models unsuitable. The authors in [14] further assume independence between window size and RTT of each window, which is inappropriate either.

In this thesis we study the interaction between TCP and link layer ARQ, examine whether implementing ARQ is beneficial, and explore the influence of protocol and loss

parameters on the overall system performance. The work is different from earlier works in several ways. First, it studies heterogenous networks where the high link loss probability and high bandwidth-delay product are essential to the system performance. Second, we focus on TCP's window flow control mechanisms and deliberately disregard other aspects of TCP, such as RTT measurements and estimation, the detail retransmission mechanism, and timer granularity. Third, the ARQ protocols considered include both GBN and SRP protocols. The delay of the ACK signals is also taken into account, which is significant for links with high bandwidth-delay product. Lastly, we give an exact analysis of the system instead of approximations and simulations, thus provide an analytical framework for future joint study of TCP and ARQ in heterogenous networks.

Specifically, the thesis investigates a system consisting of two end nodes communicating over a heterogenous network. The network includes one error-prone bottleneck link and some other link, where the error-prone link has high bandwidth-delay product. The transport layer of the two end nodes implements TCP, and the data link layer over the error-prone link implements GBN or SRP retransmission mechanism to recover its link losses.

The system is modelled as a finite state Markov chain with reward functions, and queuing models for GBN and SRP are also developed. The throughput of the system is derived by the theory of Markov chain with reward functions. The numerical results show that in most cases, implementing ARQ over the error-prone link can achieve significant improvement in system throughput. Moreover, by proper choice of protocols parameters, such as the packet size and the number of transmission attempts per packet at the link layer, significant performance improvement can be obtained.

### 1.2.2 Interaction between TCP and Random Access Scheme

The second part of the thesis work is to explore the interaction between TCP and random access by examining hybrid terrestrial-satellite networks. In such a network, a group of ground terminals share the same channel to one satellite for transmissions. There are mainly three categories of multiple access protocols: fixed assignment multiple access protocols, such as Frequency Division Multiple Access (FDMA), Time Division Multiple Access (TDMA) and Code Division Multiple Access (CDMA); random multiple access protocols (also called random access protocols), such as pure ALOHA and slotted ALOHA; and hybrid of the above two, such as reservation systems. Compared to the fixed assignment access

protocols, random access protocols have the advantage that they are simple to implement and work well for users with bursty and sporadic traffic ([12] and Chapter 2). Moreover, with random access protocols, it is easy to add and remove users as well. Hence, random access protocols are widely employed in satellite and wireless networks. This thesis considers hybrid terrestrial-satellite networks with slotted ALOHA employed at MAC layers and TCP employed at transport layers. For readers not familiar with TCP and ALOHA, a brief description is given in Chapter 2.

Since ALOHA was first proposed in 1970s, there have been many papers analyzing its performance. In [12] and the papers therein, the authors give comprehensive and rigorous analysis on several aspects of ALOHA systems, such as the throughput, collision probability and idle probability, and stability issues. These analysis considers ALOHA protocols only and disregards the effects of other layers. For idealized slotted ALOHA, one of the main assumptions these analysis make is that packets arrive at senders according to independent Poisson processes. Two of the main results on its performance are that, the maximum system throughput is $1/e$, and this throughput can be reached by adjusting the transmission probability to make the system attempt rate be one. At other attempt rate, the system has either too many idle slots or too many collisions, and the throughput is below $1/e$.

In practical networks, MAC layers receives packets from their corresponding higher layers. When a system employs TCP at its transport layer, the packets arrival process at the MAC layer is no longer a Poisson process but under the control of TCP window flow control scheme. Moreover, the maximum number of packets available at the MAC layer is limited by the current TCP window size.

In addition, from the perspective of TCP, there have been many papers studying the TCP performance in hybrid terrestrial-satellite networks [4, 50, 49, 45, 7, 27, 3, 10, 26, 40, 41]. Most of these work focuses on the properties of high bandwidth-delay product and high link loss property. The paper [43] and [36] also analyze the performance of a normal TCP session with random packet loss probability. This chapter also considers the TCP performance in hybrid terrestrial-satellite networks, but focuses on its interaction with the MAC layer random access scheme.

In particular, under random access protocols, packets enter the shared channel randomly, and collisions occur at the MAC layer with some probability. These MAC layer collisions directly affect TCP window evolution at the transport layer and hence the TCP

performance. In addition, ideal ALOHA protocol resolves collisions by MAC layer retransmissions of collided packets. When the system employs TCP at the transport layer, there is one more option, that is, resolving collisions by TCP retransmissions.

In this thesis we try to understand systems with TCP at the transport layers and ALOHA at the MAC layers, and examine the impacts of different protocols and parameters on the system performance, such as MAC layer transmission probability, TCP timeout backoff and MAC layer retransmissions. Specifically, the system investigated is a hybrid terrestrial-satellite network with a large number of identical persistent source-destination pairs. Each pair employs TCP at its transport layer and each source employs the slotted ALOHA random access scheme at its MAC layer.

An analytical model for the system considered is developed, and two simple equations are derived from which the system performance, such as the system throughput, can be obtained directly. The two equations show that the maximum possible system throughput is $1/e$, which is the same as the ideal slotted ALOHA protocol. A sufficient and necessary condition to achieve this throughput is also given. However, unlike the ideal slotted ALOHA protocol, due to the impacts of TCP at the transport layer, this condition cannot always be satisfied, and the throughput of $1/e$ cannot always be achieved.

Moreover, the impacts of system and protocol parameters on the system performance are also investigated, both analytically and numerically, which is further confirmed by simulations. The optimal MAC layer transmission probability at which the system achieves its highest throughput is derived. The results show that due to the long duration of the TCP backoff period, the throughput of systems with TCP timeout backoff is far below $1/e$. Whereas for systems without TCP timeout backoff, the results show that in cases when the round trip time is small or the number of source-destination pairs is large, a throughput of $1/e$ can be achieved by setting the MAC layer transmission probability to its optimal value. Otherwise, the maximum achievable throughput can be substantially smaller than $1/e$.

The options of TCP retransmissions and MAC layer retransmissions for collision resolution are also investigated. The analysis shows that MAC layer retransmissions react faster to collisions and hence inject more traffic into the channel. As a result, if the system operates under heavy load, for example systems with large transmission probability, large number of users, or small propagation delay, the system behaves better when retransmissions are handled by TCP.

### 1.2.3 Packet Scheduling with Window Service Constraints

Finally we consider the optimal scheduling problem with window service constraints, such as the problem of link layer scheduling subject to window service constraints that are imposed by higher layer protocols such as TCP. In traditional scheduling problems, jobs arrives at a server according to some random process. The response time of each job is defined to be the difference between its departure and arrival times. The performance of different scheduling policies is usually measured by the mean response time. It has been shown that, among all policies, for a work-conserving queue, the Shortest-Remaining-Processing-Time (SRPT) scheduling policy is optimal with respect to minimizing the mean response time [46, 48]. The author in [47] further gives the distribution of the response time for M/G/1 queue under the SRPT policy. Recently a number of papers [9, 56, 42] study the fairness property of the SRPT policy. The fairness is measured by slowdown (also called stretch), which is defined to be the ratio of the response time and the processing time of a job. They show that the SRPT policy not only minimizes the mean response time, but also is fair.

All these previous works assume that upon the arrival of a job, any part of the job is available for service. In practice, however, before being served, jobs are often broken into smaller units, and there may exist a limit on the number of units that can be served within a time interval. For example, in data networks, files are broken into messages and then packets before being released from the transport layer. If the transport layer employs TCP, then the number of packets that can be released to the lower layer is limited by the current window size of TCP, denoted by $W$. That is, at most $W$ packets can be released within one round trip time. Another example is the transmission of frames at the data link layer, where the number of frames that can be transmitted within one round trip time is limited by the window size of the data link layer protocol.

This thesis considers the optimal scheduling policy that minimizes the mean response time when there exists a limit on the number of units to be served within a fixed time interval. The limit is called window size, the units are called packets, and the constraint is called window service constraint. In particular, the effects of the window constraints are investigated in detail, and an optimal policy and a more insightful suboptimal policy are derived. The results shows that both the job lengths and the window sizes are essential to the optimal policy. Moreover, instead of changing priority of jobs at different times, in

most cases the optimal policy gives full priority to one job.

## 1.3 Thesis Overview

The rest of the thesis is organized as follows:

Chapter 2 briefly describes the three main protocols considered in this thesis: data link layer protocol ARQ, MAC layer protocol ALOHA, and transport layer protocol TCP.

Chapter 3, 4 and 5 are the core of the thesis. We first examine the interaction between TCP and ARQ in Chapter 3, then investigate the interaction between TCP and ALOHA in Chapter 4, and finally in Chapter 5 we study the problem of optimal scheduling with window service constraints.

Chapter 6 concludes the thesis.

# Chapter 2

# Protocol Descriptions

There are three protocols that will be investigated in this thesis: data link layer retransmission protocols (ARQ), MAC layer protocol ALOHA, and transport layer protocol TCP. In this chapter we briefly describe these three protocols with highlights on those aspects this thesis is focused on.

## 2.1 ARQ Protocols

### 2.1.1 Overview of ARQ Protocols

ARQ was first proposed during World War II [24, 53] to ensure reliable link transmissions between two nodes. Its basic idea is very simple: upon detecting an error/lost frame, the receiver requests the transmitter to retransmit the frame. Yet it is very efficient as well. Over the past decades, numerous advanced technologies have been developed, but ARQ remains to be the fundament of many reliable data link layer protocols. In this section, we briefly describe its basic operations. Readers interested in details can refer to [12, 35, 38, 51, 57].

ARQ protocols guarantee reliable transmissions by retransmitting error/lost frames. Different versions of ARQ protocols have different ways to handle transmissions and retransmissions. In general they use some or all of the following techniques:

1. *Positive acknowledgment and transmission:* Upon receiving an error-free frame, the receiver sends an ACK back to the transmitter. This ACK may allow the transmitter to transmit more frames.

2. *Negative acknowledgement and retransmission:* Upon receiving an error frame, the

receiver sends a negative acknowledgment (NAK) back to the transmitter. This NAK requests the transmitter to retransmit the corresponding frame.

3. *Retransmission after timeout:* The transmitter sets a timer for each frame transmitted. If a frame has not yet been acknowledged upon its timer expires, the transmitter retransmits this frame.

Note that feedback frames containing ACKs and NAKs may also incur errors or losses.

There are three basic types of ARQ protocols: the Stop-and-Wait protocol, the Go-Back-n (GBN) protocol, and the Selective-RePeat (SRP) protocol. The former the simplest and the latter the most efficient. The Stop-and-Wait protocol is a special case of the GBN protocol with the parameter $N = 1$, and the GBN protocol is essentially a sliding window protocol. In the following subsections we first describe sliding window protocols, then discuss the operations of the three types of ARQ protocols in detail, where all techniques listed above will be discussed.

### 2.1.2 Sliding Window Protocols

In sliding window protocols, only packets within the window can be transmitted. Packets are hence classified into four groups: packets that have been sent and acknowledged, packets that have been sent but not acknowledged yet, packets that have never been sent before but can be sent as soon as possible, and packets that have not been sent yet and cannot be sent either until the window moves. Figure 2-1 illustrates the window and the four groups.

Specifically, in Figure 2-1, the window spans from packet 4 to packet 9. Within the window, part of the packets have been sent but not acknowledged yet, which are packet 4, 5 and 6, while the rest packets are packets that have never been sent before but can be sent as soon as possible, which are packet 7, 8 and 9. Outside the window, on the left are packets that have been sent and acknowledged, which are packets 1, 2 and 3, and on the right are packets that have not been sent yet and cannot be sent either until the window moves rightward and they fall into the window, which are packets after 10. A new ACK received will increase the number of packets that have been sent and acknowledged and move the left edge of the window rightwards. If the window size is fixed such as that in the GBN protocol, this movement allows more packets on the right to fall into the window and become available to send as soon as possible. The name 'sliding window protocol' comes

from this window 'sliding' behavior.

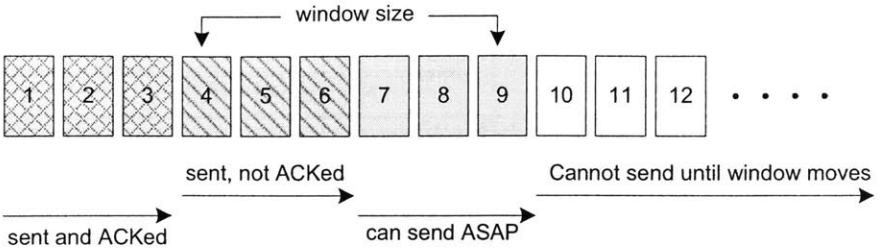


Figure 2-1: Illustration of Sliding Window Protocols

Note that here we implicitly assume that the ACKs arrives in order. As will be shown later, this is true with the GBN protocol, but not with the SRP protocol.

### 2.1.3 The Stop-and-Wait Protocol

The Stop-and-Wait protocol is the simplest type of ARQ protocols. Under this protocol, no new frames will be transmitted until the previous one has been received correctly. Hence the system has at most one frame in transmission.

Using the techniques described in Section 2.1.1, Figure 2-2 shows an example of how the Stop-and-Wait protocol handles transmissions and retransmissions upon error-free frames, error frames, lost frames and lost ACKs. Specifically, after receiving the error-free packet 1, the receiver sends back an ACK to the transmitter. This ACK allows the transmitter to transmit a new packet, that is packet 2. Whereas upon detecting the error frame containing packet 2, the receiver sends back a NAK, which requests the transmitter to retransmit packet 2. In addition, both the lost frame containing packet 3 and the lost ACK for the first transmission of packet 4 result in timer expirations and retransmissions of the corresponding packets.

Since the Stop-and-Wait protocol allows only one frame in transit, no more packets can be transmitted while waiting for an ACK/NAK/timer-expiration. Hence the link utilization is highly inefficient. The GBN protocol and the SRP protocol improve the link utilization by allowing more packets in transit.

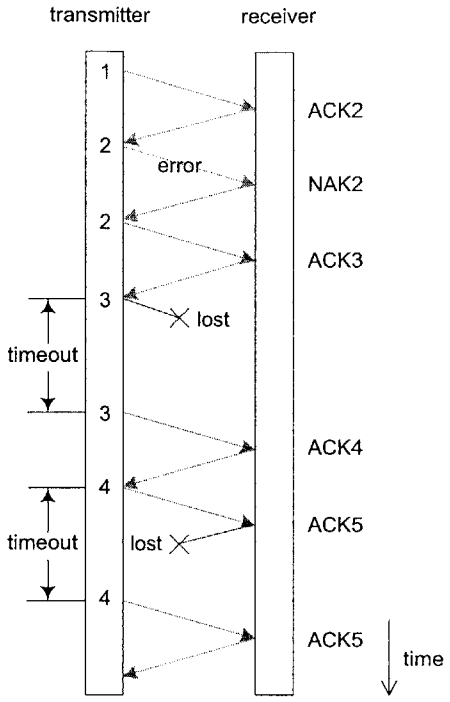


Figure 2-2: Example of the Stop-and-Wait protocol

### 2.1.4 The GBN Protocol

The GBN protocol is essentially a sliding window protocol with fixed window size $N$. Different from the Stop-and-Wait protocol, while waiting for ACK/NAK/timer-expiration, in the GBN protocol the transmitter is allowed to transmit up to $N$ successive packets. The link utilization is hence greatly improved.

However, in the GBN protocol, the receiver has no buffer. Since there are multiple packets in transit, the receiver may receive out-of-order packets. Upon receiving out-of-order packets, the receiver discards the out-of-order packets and sends back a NAK to request a retransmission of the packet it is waiting for.

Figure 2-3 illustrates the operation of the Go-Back-3 protocol upon error-free frames, error frames, lost frames and lost ACKs, with the sliding window shown at the left. Specifically, at the beginning one window of packets are transmitted in order, that is packet 1 to 3. Each time after the receiver receives an in-order error-free packets, the receiver sends back a new ACK to the transmitter, that is ACK2 to ACK9. Correspondingly, each time after

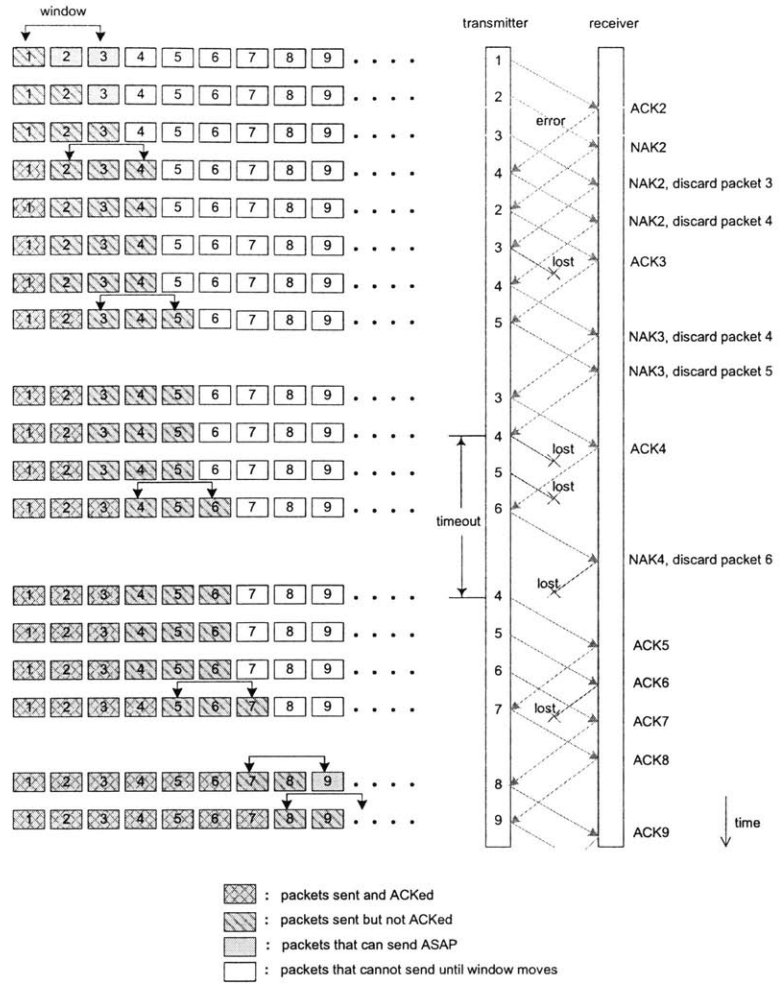


Figure 2-3: Example of the Go-Back-N Protocol

the transmitter receives a new ACK, the window slides rightwards and allow more packets to be available to send as soon as possible.

In addition, although the first reception of packet 3 and 4 are successful, they are out-of-order packets. The receiver hence discards them and sends back NAK2 to request packet 2 it is waiting for.

Moreover, upon detecting the error frame containing packet 2, the receiver sends back NAK2 to the transmitter, which in turn retransmits packet 2. The transmitter then continues transmits successive packets, that is packet 3 and 4, until the window is used up.

Figure 2-3 also shows that lost packets can be retransmitted either upon a NAK or a timer-expiration. For example, lost packet 3 is retransmitted upon NAK3, which was

generated by out-of-order error-free frame containing packet 4. For lost packet 4 in Figure 2-3, neither ACKs nor NAKs are received by the transmitter before its associated timer expires. This timeout triggers the retransmission of packet 4.

Furthermore, we can see from Figure 2-3 that ACKs in GBN are cumulative. For example, although ACK6 is lost, ACK7 received by the transmitter indicates that packet 5 has already been correctly received. Consequently, the transmitter and the receiver moves on to transmit successive packets.

From Figure 2-3 we can see that once one packet is retransmitted, all its successive packets up to one window are retransmitted. In other words, all previous transmissions of these packets are useless, no matter they are error or error-free. The GBN protocol can hence be modelled as a queuing system, with packet service time to be the time between its first transmission and its successful transmission [12, 60, 54, 34].

The above discussions on the Stop-and-Wait protocol and the GBN protocol shows that the Stop-and-Wait protocol is actually a special case of the GBN protocol with N=1. On one hand, because $N = 1$, in the Stop-and-Wait protocol there are no out-of-order packets issue, and the operation becomes much more simpler. On the other hand, N=1 makes the transmitter be unable to transmit more packets while waiting for an ACK/NAK/time-expiration, which results in a waste of link capacity.

Note that in the GBN protocol, since the receiver has no buffer, all out-of-order packets are discarded and will be retransmitted even if they are received correctly. This also leads to a waste of link capacity. The SRP protocol allows the receiver to have buffer, which reduces these unnecessary retransmissions and further improves the link utilization.

### 2.1.5 The SRP protocol

The main difference between the GBN protocol and the SRP protocol is that in the SRP protocol, the receiver has buffer. The out-of-order packets can then be stored in the buffer, and the unnecessary retransmissions are thus avoided.

Specifically, upon receiving an out-of-order packet, the receiver stores this packet in its buffer and sends back to the transmitter an ACK that acknowledges this packet. Once the missing packets arrive at the receiver, the receiver will deliver a batch of packets in order to the higher layer.

Note that unlike the GBN protocol, in the SRP protocol out-of-order packets no longer

generate NAKs to request missing packets. As a result, a lost packet will eventually lead to a timeout. In addition, since out-of-order receptions are possible, ACKs no longer arrive at the transmitter in order and are not cumulative anymore. Consequently, a lost ACK or NAK will also lead to a timeout. Moreover, out-of-order receptions also lead to the result that some packets within the window may be packets that have been sent and acknowledged. Hence the SRP protocol is no longer a sliding window protocol.

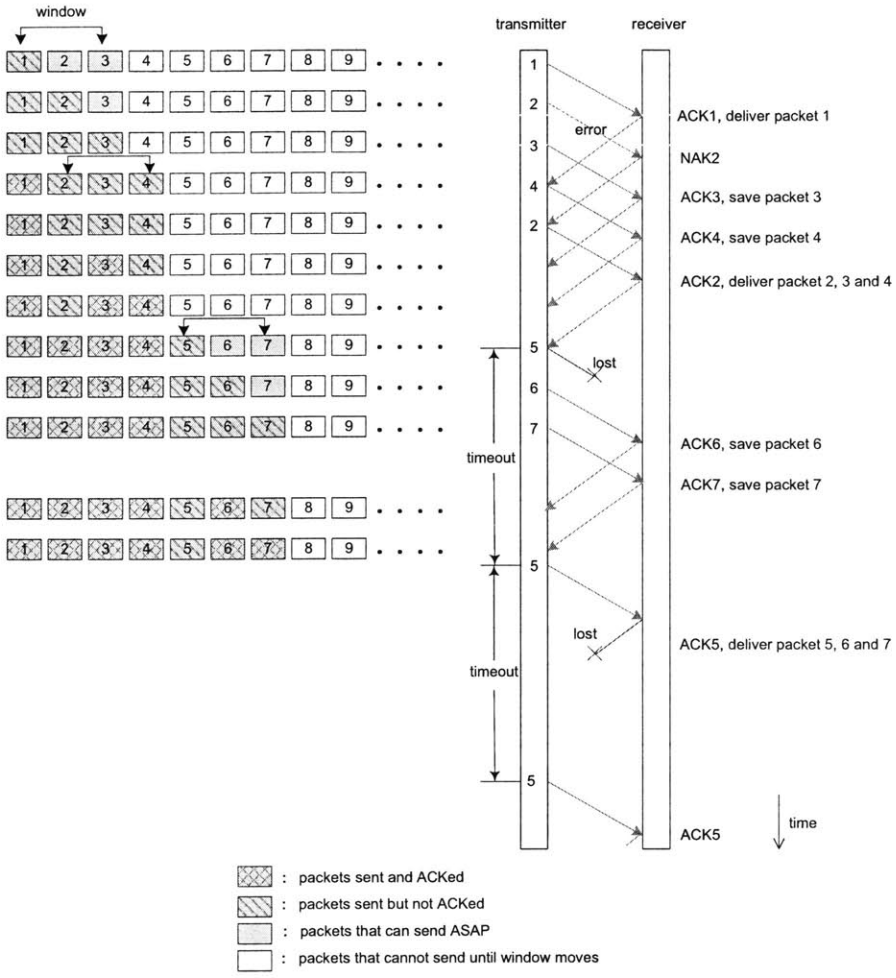


Figure 2-4: Example of the Selective RePeat Protocol

Figure 2-4 gives an example of the SRP protocol with window size of 3. The window is shown at the left. As shown in the figure, out-of-order packets (packet 3, 4, 6 and 7) are saved at the receiver buffer until the missing packets (packet 2 and 5) arrive. At this point batch of packets are delivered to the higher layer (packet 2, 3 and 4 together and packet 5,

6 and 7 together).

Figure 2-4 also shows that the error-frame containing packet 2 results in a NAK, which further leads the transmitter to retransmit packet 2. Whereas both lost packet 5 and lost ACK5 result in timeouts and retransmissions of packet 5 by the transmitter.

By investigating the window shown at the left of Figure 2-4, we can see that here within the window, there are not only packets that have been sent but not acknowledged and packets that have not been sent yet but are available to send as soon as possible, but also packets that have been sent and acknowledged. As mentioned previously, this is due to the out-of-order packets allowed in the SRP protocol.

For convenience, define packet service time in the SRP protocol to be the time between its first transmission and the reception of the ACK that acknowledges its successful transmission. From Figure 2-4 we can see that unlike the GBN protocol, in the SRP protocol, the retransmission of one packet does not lead to retransmissions of its successive packets. Although physically the system is still a queuing system, the service times of different packets overlap with each other. Nevertheless, the transmissions of different packets are independent, in the sense that transmissions of one packet depend only on the previous transmissions of the same packet, and totally independent of the status of other packets. Many models of the SRP protocol is based on this independent property.

### 2.1.6 Discussions

This section briefly describes the three main types of ARQ protocols: the Stop-and-Wait protocol, the GBN protocol and the SRP protocol. If each frame incurs a loss independently with probability $p$ and the round trip delay is $D$, it can be shown that the throughput, denoted by $\eta$, of the three protocols are: $\eta_{Stop-and-Wait} = (1-p)/D$, $\eta_{GBN} = (1-p)/(1+pD)$ and $\eta_{SRP} = 1-p$, respectively [12, 51].

For these three protocols, the authors of books [12, 35, 38, 51, 57] give the details of their algorithms at the transmitter and receiver and rigorous proofs of the reliability, as well as some other aspects such as framing. In addition, the authors in [34, 54, 60] developed queuing models for these protocols and derived the expected queue length and average delay. Currently queuing analysis is one of the main mathematical tools to analyze ARQ protocols.

## 2.2 MAC Layer Protocol ALOHA

### 2.2.1 Overview of MAC Layer Protocols

Multiple access issue emerges when there are multiple transmitters sending data to one receiver. In this case system resources must be divided among all transmitters. Currently there are three main categories of multiple access protocols: fixed assignment multiple access, random multiple access (also called random access) and hybrid of the above two.

Fixed assignment multiple access protocols include Frequency Division Multiple Access (FDMA), Time Division Multiple Access (TDMA), Code Division Multiple Access (CDMA) and their variations and modifications. Random access protocols include ALOHA [2] and its variations and modifications, such as splitting algorithm [15] and Carrier Sense Multiple Access (CSMA) [33]. Reservation systems and spread ALOHA are examples of hybrids of fixed assignment and random multiple access. In general, fixed assignment multiple access protocols divide the resources into $m$ portions, where $m$ is the number of users, and allocate one portion to each user. Whereas random access protocols allow users to access the full resource at the expense of collisions.

Different multiple access protocols have different properties and are suited for different scenarios. If we view the multiple access system as a queuing system, it can be shown that in terms of average and variance of delay, fixed assignment protocols such as FDMA and TDMA work well for regular users, and random access protocols such as ALOHA are good to users with bursty but sporadic traffic [12]. One limitation of fixed assignment protocols is that it is more difficult to add or remove users by implementing them than that by implementing random access protocols. One limitation of random access protocols is the collision problem that can be solved by retransmissions.

There are two types of random access protocol ALOHA: pure ALOHA and slotted ALOHA. The main difference between them is that in slotted ALOHA, time is divided into slots and the system becomes discrete time systems, while in pure ALOHA, slots play no role. The resulting performance is different as well. Slotted ALOHA will be part of the subjects of this thesis. The following section describes its main features in detail.

### 2.2.2 Slotted ALOHA Protocol

In idealized slotted multiple access system [12], there are $m$ senders and one receiver. All transmitted packets are assumed to have the same length and require one time unit, called slot, for transmission. All senders are synchronized. In addition, packets are transmitted in the following way: each new arrived packets are transmitted immediately. Each packet involved in a collision will be retransmitted in the next slot with some probability. This retransmission continues until the packet is successfully received. A complete description of the idealized slotted multiple access system can be found in [12].

It has been shown that with slotted ALOHA protocol, the system throughput, denoted by $\lambda$, is also the probability of a successful transmission in a slot. Moreover, the throughput can be expressed as follows:

$$\lambda = G(n)e^{-G(n)}, \tag{2.1}$$

where $G(n)$ is the attempt rate, defined to be the expected number of attempted transmissions in a slot when the number of backlogged senders is $n$.

Similarly, the probability of an idle slot, denoted by $P_I$, can be shown to be:

$$P_I = e^{-G(n)}. \tag{2.2}$$

The above equations shows that the number of packets transmitted in a slot is well approximated as a Poisson random variable. The maximum possible throughput is 1/e, which can be achieved when $G(n) = 1$.

Further analysis on the system also shows that when $m$ goes to infinity, the system is unstable. A great amount of work has been done on stability issues, estimation of number of backlogged senders, as well as the way to set the retransmission probability to achieve maximum throughput of 1/e. These issues are beyond the scope of this thesis. Interested readers can refer [25, 37, 55, 12].

## 2.3 Transport Layer Protocol TCP

### 2.3.1 TCP Overview

As mentioned in Chapter 1, the main function of transport layer in data networks is to perform congestion control and provide reliable end-to-end message transmission if required by higher layers. Currently TCP is the dominate reliable transport layer protocol in data networks [52]. For a reliable transport layer protocol, generating and receiving acknowledgements (ACKs) by the transmission nodes is essential to guarantee reliable transmissions. To achieve better throughput while performing congestion control, it is also desirable for the end nodes to maximize the utilization of the available bandwidth provided by the network. The beauty of TCP is that it can simultaneously provide reliable transmissions and explore the available bandwidth in the network without requiring any additional information beyond the necessary ACKs.

Specifically, TCP guarantees reliable transmissions by retransmitting lost packets and performs congestion control as well as exploring available bandwidth by window flow control. Different TCP versions have different mechanisms for retransmissions and window flow control. The most popular two TCP versions are TCP Tahoe and TCP Reno. The following section describes their retransmission and window flow control mechanisms.

### 2.3.2 TCP Retransmission and Window Flow Control Mechanisms

Packet transmissions of both TCP Tahoe and TCP Reno follow sliding window protocol (see Section 2.1). The window size is determined by their window flow control schemes. Upon a loss, both TCP Tahoe and TCP Reno retransmit the first lost packet detected and resume sliding window protocol from this packet. The way TCP Tahoe detects losses is the timeout signal. TCP Reno has one more way to detect losses, that is duplicate ACKs. Timeout signal and duplicate ACKs are also signals that trigger window close/shrink in TCP Tahoe and Reno, as described later in their window flow control mechanisms.

For TCP Tahoe and Reno window flow control mechanisms, this thesis works on the following simplified versions: both TCP Tahoe and TCP Reno keep two parameters: current window size $W$ and window threshold $W_t$. In addition, based on its available buffer size, TCP at the receiver side also announces a window size called receiver's advertised window size $W_M$. TCP Tahoe window flow control algorithm works as follows:

- The initial window size is 1.
- Upon an acknowledgment:
  - If $W < W_t$, then $W \leftarrow W + 1$;
  - If $W \geq W_t$, then $W \leftarrow W + 1/\lfloor W \rfloor$.

  If the resulting window size $W > W_M$, then $W \leftarrow W_M$.
- Upon a loss signal, $W_t \leftarrow W/2$ and $W \leftarrow 1$.

TCP Reno window flow control algorithm consists of the first two of the above rules that TCP Tahoe algorithm has and the following addition rule:

- Upon a timeout signal, $W_t \leftarrow W/2$ and $W \leftarrow 1$.

  Upon three duplicate ACKs, $W_t \leftarrow W/2$ and $W \leftarrow W_t$.

In both mechanisms, the phase with $W < W_t$ is called slow start phase, and the phase with $W \geq W_t$ is called congestion avoidance phase. Figure 2-5 illustrates the window evolution in these two phases.

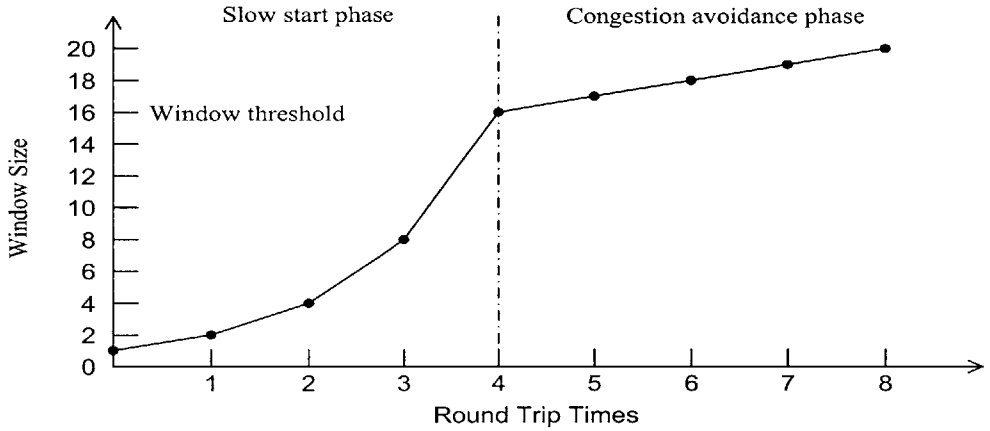


Figure 2-5: Illustration of TCP Window Evolutions

Notice that the actual TCP Tahoe and Reno window flow control mechanisms include many details. For tractability and clarity, the above simplified versions disregard those details but extract their essential parts. The above versions are also versions widely accepted and used by other researchers working on TCP [36].

Also notice that since what we are interested in are TCP's retransmissions and congestion control, we deliberately disregard other TCP issues, such as connection establish and termination, and time granularity. For interested readers, the details of TCP retransmission and window flow control and other aspects can be found in Steven's book [52].

### 2.3.3 Observations from TCP Operations

From the above retransmission and window flow control mechanisms, we have the following observations:

1. TCP window size determines how many packets of this connection can be injected into the network within one round trip time (RTT). Therefore, together with RTT, TCP window size directly determines the send rate of this connection, which is closely related to TCP performance.

2. TCP window size is also the maximum number of outstanding packets of this connection. Smaller window size means less packets of this connection in the network and vice versa. Window size is thus directly related to the congestion level of the network. This is the fundament how TCP performs its congestion control, that is, by regulating its window size.

3. TCP window flow control mechanisms say that the only moment when the window size can be increased is when an ACK is received. In slow start phase, the window size is increased by one upon each ACK. The increase rate is actually very high (contrary to its name. See Figure 2-5). Whereas in congestion avoidance phase, the window size is increased by one upon $W$ ACKs, or per RTT. The increase rate becomes slow. The reason behind this is that TCP interprets its send rate corresponding to the window threshold $W_t$ as compatible with the available bandwidth provided by the network. Hence when the window size is below $W_t$, it increases the window size very fast up to this level to fully utilize the available bandwidth; while after the window size reaches $W_t$, it increases its window size slowly to explore extra bandwidth available without causing sudden congestion.

4. TCP window flow control mechanisms also say that upon a loss (timeout signal in both Tahoe and Reno and duplicated ACKs in Reno only), TCP closes/shrinks its

window size. Recall that the purpose of TCP window flow control is for congestion control. That is, TCP regards all losses as congestion losses and performs congestion control correspondingly. Saying it in another way, each loss signal not only triggers retransmission of lost packets, but also leads to window close/shrink.

5. The way TCP Reno reacts the two type of loss signals, timeout signal and duplicate ACKs, are different. When timeout occurs, TCP interprets it as that the network is heavily congested, and greatly reduced its window size to 1. Whereas when duplicate ACKs are received, there are still packets getting through the network. TCP thus interprets it as that although there are some congestion in the network, the level is light. Accordingly, TCP reduces its window size to the window threshold $W_t$, and restarts exploring extra bandwidth in its congestion avoidance phase.

6. TCP window flow control has Markov property. That is, given current window size $W$ and window threshold $W_t$, the future window evolution is independent of its past. In many cases, this property allows to model the system considered as a Markov chain. However, when the window size $W$ can be big, the state space could be very large.

7. TCP has self-clocking behavior. That is, the spacing of ACKs returned to the sender is identical to the spacing of packets [52] (In reality they are different due to different queuing delay and path packets go through).

The above observations are important for the later discussion on TCP interaction with other protocols, as we will see later in Chapter 3 and 4. They also provides us some intuition on why TCP may not perform well in heterogenous data networks. For example, observation 3 says that the window increase rate in congestion avoidance phase is slow. For a heterogenous data networks with high bandwidth-delay product, once TCP jumps into the congestion avoidance phase with small window size, this slow increase rate leads to a waste of significant amount of available bandwidth. Another example is that observation 4 says that TCP regards all losses as congestion losses and reduces window size accordingly. For a heterogenous data networks with high link loss probability, the link losses will also be interpreted as congestion losses and TCP will 'falsely' reduce its window size. As a result, the available bandwidth is significantly under-utilized. For a heterogenous data networks with both high bandwidth-delay product and high link loss probability, such as satellite

networks, the above discussions show that fundamentally TCP alone cannot perform well. We will have comprehensive discussions on these issues later.

### 2.3.4 TCP Timeout Value

One last thing about TCP we would like to discuss here is its timeout value. TCP timeout value, used to generate the timeout signal and denoted by *RTO*, is updated upon RTT measurements. The details about update rules can be found in [28, 52]. Here we would like to point out its exponential backoff and Karn's algorithm [30]. The former considers successive timeouts, and the latter considers *RTO* update upon retransmissions.

After a timeout and before an acknowledgement is received for a packet that was not retransmitted, *RTO* is doubled each time a timeout occurs until it is 64 times of its original value or it is more than 64sec. This is called exponential backoff. The reason behind to do so is that TCP interprets multiple timeouts as that the network is heavily congested. Accordingly TCP greatly reduces the send rate by this exponential backoff to relieve the congestion level. Figure 2-6 illustrates this timeout exponential backoff, where the white boxes indicate successful transmissions and the gray boxes indicate unsuccessful transmissions. In the figure, packet 2 incurs a timeout three times and is retransmitted three times. No ACK is received between these timeouts. Hence *RTO* is doubled each time a tiemout occurs.

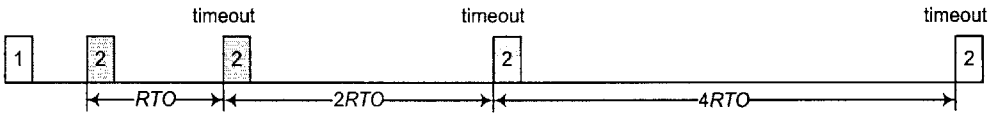


Figure 2-6: Illustration of TCP Timeout Backoff

The purpose of Karn's algorithm is to resolve so called retransmission ambiguity problem. When an acknowledgement of a packet that has been retransmitted before is received, we do not know if this ACK is for the original transmission or for one of its retransmissions. Karn's algorithm specifies that the RTT estimates and *RTO* should not be updated upon this ACK. In addition, since the packet is retransmitted, the exponential backoff is applied to *RTO*. This backed off *RTO* is reused until an ACK for a packet that has never been retransmitted is received.

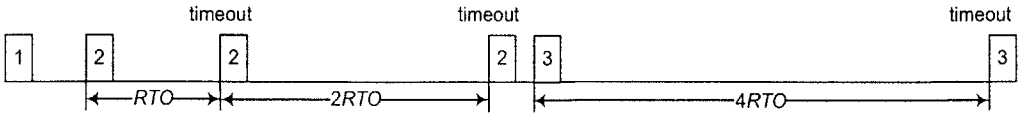


Figure 2-7: Illustration of Karn's Algorithm

Figure 2-7 illustrates Karn's algorithm. In this figure, packet 2 is transmitted three times in total, and the third transmission is successful. The corresponding ACK triggers the transmission of packet 3. However, we do not know which of the three transmissions the ACK is for. Therefore, instead of being updated by this ACK, *RTO* continues to be backed off. This backed off *RTO* is used when packet 3 incurs a timeout. Note that although there are successful transmissions (the third transmission of packet 2 in Figure 2-7), *RTO* may still be backed off.

# Chapter 3

# Interaction between TCP and ARQ

This chapter considers interaction between TCP and ARQ in networks with high link loss probability and high bandwidth-delay product. We will first introduce the background of this topic, then describe in detail the system under consideration and develop analytical model for it. Finally, we will discuss the numerical results for different protocol and packet loss parameters, and give our conclusions on this subject.

## 3.1 Background

Currently TCP is the dominate reliable transport layer protocol in data networks [52]. As such, the beauty of TCP is that it can simultaneously provides reliable transmission and explore the available bandwidth in the network without requiring any additional information beyond the necessary ACKs. Specifically, TCP makes use of the rate and order of ACKs to regulate its window size, thus its transmission rate, to perform congestion control and explore the available bandwidth [52].

TCP works very well in classical wireline networks, as evident by the current Internet. Unfortunately, TCP does not perform well in heterogenous data networks, especially those with high link loss probability and high bandwidth-delay product [4, 50, 49, 45]. There are two major reasons for TCP's poor performance in such heterogenous data networks. One is that TCP regards all losses as congestion losses and reduces window size accordingly. For data networks with high link loss probability, the link losses will thus lead TCP to 'falsely' reduce its window size, which further results in low transmission rate and significant waste of link capacity. Another reason is that the window increase rate in TCP congestion avoidance

phase is slow. For data networks with high bandwidth-delay product, this slow increasing rate takes TCP a long time to raise its window size comparable to the available network capacity, which lead to significant waste of the capacity. For networks with both high link loss probability and high band-width delay product, it is foreseeable that with high probability, the frequent window shrinks/closes due to link losses plus the slow window increase rate result in TCP operating at a window size far below the available network capacity. As a result, the overall TCP performance is significantly deteriorated.

Many solutions are proposed to enhance TCP performance over this type of heterogenous networks, such as window scaling, time stamps, selective acknowledgements, TCP spoofing, split-TCP, and link layer solutions. Comprehensive descriptions of the above solutions can be found in [7, 27, 45, 3, 10]. Among all these solutions, link layer ARQ has the advantage that it fits naturally into the layered structure of networks. The main idea of implementing ARQ is that it can "hide" the link losses from TCP. In this sense, implementation of ARQ can greatly reduce the number of unnecessary TCP window shrinks/closes and significantly improve the overall TCP performance. However, the presence of an ARQ protocol can lead to other more subtle problems. In particular, due to the high bandwidth-delay product, ARQ retransmissions introduce high variability of the packet round trip time (RTT) seen by TCP, and lead to TCP timeouts due to ARQ retransmissions. We call these timeouts **false timeouts**.

Recently a number of papers have examined the interactions between transport layer TCP and link layer ARQ protocols. Most of them involve simulations in wireless environment [7, 19, 20, 39, 58, 5, 13]. There is some analytical work as well [14, 16, 17, 18, 43]. These papers provide approximate analysis of the system performance with transport layer TCP and a link layer protocol with the assumption of instantaneous ACK feedback. The large bandwidth-delay product of heterogenous networks we consider here makes these models unsuitable for our scenario. The authors in [14] further assume independence between window size and RTT of each window, which is not suitable for our scenario either. In addition, there are many papers on the analysis of ARQ as well. In [54] queuing models are developed for the Go-Back-N (GBN) protocol, and [34] provides queuing models for both the GBN and Selective-Repeat (SRP) protocols. Generally queuing models are used as a tool to analyze the ARQ protocols for different channels and network structures [60, 63].

This chapter studies the interaction between TCP and link layer ARQ in the context of

heterogenous networks with high link loss probability and high bandwidth-delay product, examines whether implementing ARQ is beneficial, and explores the influence of protocol and loss parameters on the overall system performance. This work is different from earlier works in several ways. First, it studies heterogenous networks where the high link loss probability and high bandwidth-delay product are essential to the system performance. Second, we focus on TCP's window flow control mechanisms and deliberately disregard other aspects of TCP, such as RTT measurements and estimation, the detail retransmission mechanism, and timer granularity. Since the window size evolution is the main factor that affects TCP throughput, we believe that for the purpose of investigating TCP throughput, the protocols considered capture the essence of TCP. Third, the ARQ protocols considered include both GBN and SRP protocols. The delay of the ACK signals is also taken into account, which is significant for links with high bandwidth-delay product. Furthermore, we give an exact analysis of our system instead of approximations and simulations, thus provides an analytical framework for future joint study of TCP and ARQ in heterogenous networks.

Specifically, the system investigated consists of two end nodes communicating over a heterogenous network that includes one error-prone bottleneck link and some other links. The error-prone link has high bandwidth-delay product. One example of such networks are hybrid space-terrestrial network, where the satellite link corresponds to the error-prone link. The sender has unlimited number of packets to be sent to the receiver, and the performance metrics considered is the throughput. The losses incurred by packets include the link losses of the error-prone link and other random losses, for example, losses of other links and congestion losses. The error-prone link implements GBN or SRP retransmission mechanism to recover its link losses. The transport layer of the end nodes implements a variation of the Additive-Increase-Multiplicative-Decrease (AIMD) protocol that is similar to TCP window flow control protocols, which will be described in detail later.

We model the system as a finite state Markov chain with reward functions. The transition probabilities and the reward functions are expressed in simple window-based product form and sum form, respectively. Moreover, queuing models for GBN and SRP are also developed. These models are used to obtain the probabilities needed for solving the transition probabilities and the reward functions of the Markov chain, and the throughput of the system is derived by the theory of Markov chain with reward functions. The numerical

results show that in most cases implementing ARQ over the error-prone link can achieve significant improvement in system throughput. Moreover, we show that by proper choice of protocols parameters, such as the packet size and the number of transmission attempts per packet at ARQ, significant performance improvement can be obtained.

The chapter is organized as follows: Section 3.2 describes in detail the system under consideration. Section 3.3 and 3.5 model different systems considered, and Section 3.4 provides the simple queuing models for the ARQ protocols. Finally, Section 3.6 discusses the numerical results for different protocol and packet loss parameters, and Section 3.7 concludes the chapter.

## 3.2 System Description

The system we consider consists of two end nodes communicating over a heterogenous network with one error-prone link and some other links, as shown in Figure 3-1. The sender has unlimited packets to be transmitted to the receiver, and these packets have fixed length. The transport layer (TL) at the end nodes implement a variation of AIMD protocol, which is similar to TCP window mechanisms and will be described in detail later. The link layer over the error-prone link, called (ARQLL), implements an ARQ protocol, where both GBN and SRP are considered. The other link layers (LLs) do not employ ARQ.

The error-prone link in the system is the bottleneck link and has long bandwidth-delay product. The time for the error-prone link to transmit one packet is defined to be one time unit. In this way, the time is divided into time slots. The round trip delay of one transmission over the error-prone link, defined to be the interval between the time the error-prone link sender sends out a packet and the time the sender receives the acknowledgment of this transmission, is fixed to be $d$ time slots. The remaining time needed for the packets to go through the network is assumed to be negligible, a reasonable assumption since the error-prone link has long bandwidth-delay product.

Packets in the system incur two types of losses. One type refers to losses over the error-prone link. For brevity, we call those **link losses**. The other type refers to all the losses other than the error-prone link losses, such as congestion losses and link losses at the other links that do not employ ARQ. We call those **random losses**. Note that the link losses can be recovered by the employed ARQLL ARQ, and the random losses can only be recovered

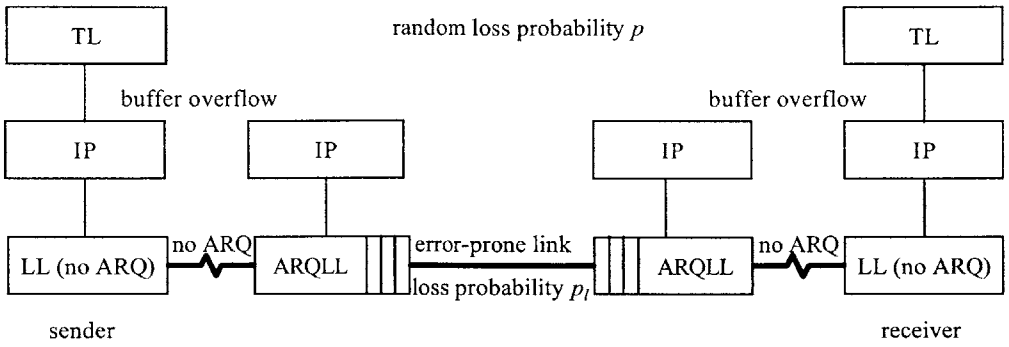


Figure 3-1: System with Two Nodes Communicating over a Heterogenous Network

by the end-to-end transport layer retransmissions.

Packets incur losses independently of each other. Each packet incurs a random loss with probability $p$, and each transmission over the error-prone link incurs a link loss with probability $p_l$. The probability that a packet incurs no random loss and a transmission over the error-prone link incurs no link loss, denoted by $q$ and $q_l$, respectively, are thus $q = 1 - p$ and $q_l = 1 - p_l$. All acknowledgements (ACKs), including both the transport layer ACKs and the ACKs of ARQ, are assumed to be loss free. Notice that by proper choice of the above loss probabilities, one can also model systems employing no ARQLL ARQ.

The TL has two ways to detect losses, the **timeout signal** and the **random loss signal**. When the age of an unacknowledged packet exceeds the TL timeout value, denoted by $TO$, a TL timeout occurs. We also assume that any random loss can be detected by the node where the loss happens after a fixed time interval $t_d$. After detecting the loss, the node will generate a signal indicating this loss and send it via the same path as that of a packet to the receiver. We call this signal the random loss signal. The timeout signal is essential for reliable TL transmissions [61], and the random loss signal is used to approximate the duplicate ACKs used in TCP and allows us to examine the fast retransmit and fast recovery mechanism.

The AIMD protocol employed at the TL works as follows. The TL sends packets in batches. The size of the batches is the current window size of the TL. Once the TL receives all the acknowledgments of its previous window of packets, it sends out the next window of packets. After either a timeout signal or a random loss signal is received, the TL changes the window size according to its window-update algorithm, and restarts the transmission

from the packet that incurs the loss signal.

We consider two window-update-algorithms, the Tahoe window update algorithm and the Reno window update algorithm [52] 2.3, corresponding to TCP Reno and Tahoe respectively. Both algorithms have two parameters: current window size $W$ and a threshold $W_t$. In addition, there is a maximum window size $W_M$ that the window size can never exceed, which resembles the maximum window size in TCP advertised by the receiver. The Tahoe algorithm works as follows:

- The initial window size is 1.
- After receiving all the acknowledgments of the last window:
  - If $W < W_t$, the window size is doubled;
  - If $W \geq W_t$, the window size is increased by 1.

  If the resulting window size is larger than $W_M$, set the window size to be $W_M$; otherwise, keep it.
- Upon a timeout signal, the window size is set to be 1, and $W_t$ is set to be half of the window size when the timeout occurs.

The Reno algorithm consists of all the above rules that the Tahoe algorithm has and the following addition rule:

- Upon a random loss signal, $W_t$ is set to be half of the window size when the loss signal is received, and the window size is set to be $W_t$.

One can see that both algorithms are variations of Additive-Increase-Multiplicative-Decrease (AIMD) algorithms. They are similar to the TCP Tahoe and Reno window-update algorithms, respectively. The two phases, $W < W_t$ and $W \geq W_t$, correspond to the slow start phase (SS phase) and the congestion avoidance phase (CA phase) of TCP, respectively. For convenience, we also call these two phases the SS phase and the CA phase. The maximum window size $W_M$ corresponds to the receiver advertised window size. The differences between the two algorithms and the TCP Tahoe and Reno lie in two aspects. One is that in TCP the window size is updated upon receiving each acknowledgment, while here the window size is updated when a batch of acknowledgments is received. Nevertheless,

upon receiving a loss signal, the rates of window update are the same for TCP and our corresponding algorithms. The other difference is that in the TCP Reno fast retransmit and fast recovery is triggered by duplicate ACKs, while in our Reno window update algorithm fast retransmit and fast recovery is triggered by random loss signals. The random loss signals are used to simplify the analysis of the protocols. Note that our goal is to explore whether or not implementing ARQ at the error prone satellite link will improve the TCP performance, and the key factor is the window size. We therefore believe that our model is a reasonable model for this purpose and can give us useful insights of the system performance.

The ARQLL employs the standard GBN or SRP [12] 2.1. The GBN or SRP window size is no less than the round trip delay $d$, so that the error-prone link capacity can be fully utilized and the actual transmission rate over it is limited only by the TL window size. Furthermore, negative acknowledgement (NAK) signals are assumed to be used. The order of packet transmissions follows the standard GBN or SRP rules and packets are delivered to the corresponding higher layer in order [12].

In the next two sections, we will first model the system with the Reno algorithm and ARQ (both GBN and SRP), and then derive the system throughput as a function of the protocol and loss parameters. This model can be easily extended to systems with the Tahoe algorithm and sytems with some variations of ARQ, which will be shown in Section 3.5.

## 3.3 Modelling the System with Reno Algorithm

### 3.3.1 System Markov Model and Throughput

Consider the system behavior after the TL receives a loss signal. Let $W^e$ denote the window size upon the loss signal and $S$ indicate the type of loss signal received, with $S = TO$ referring to a timeout signal and $S = RL$ referring to a random loss signal. Let $W$ and $W_t$ denote the window size and threshold right after the loss signal. Then, according to the Reno window update algorithm, $W_t = W^e/2$ and if $S = TO$, $W = 1$ or if $S = RL$, $W = W^e/2$ . This means that the pair $(W^e, S)$ uniquely determines $W$ and $W_t$. Moreover, by the algorithm, $W$ and $W_t$ together are the state of the system, that is, given $W$ and $W_t$, the future behavior of the system is completely determined and is independent of its past. Therefore, the pair $(W^e, S)$ is also a state of the system. The system can thus be modelled by the following Markov chain: the states are the $(W^e, S)$ pairs, and the transitions take

place when the TL receives a loss signal. Furthermore, since the window size will never exceed the maximum value $W_M$, the system chain is a finite state Markov chain.

Figure 3-2 (a) and (b) plots the transitions from a typical state $i$ with $S_i = TO$ and $S_i = RL$, respectively. Here the subscript $i$ denotes the state. Later in the next subsection, we will explore in detail the system behavior during one transition and derive the transition probabilities.

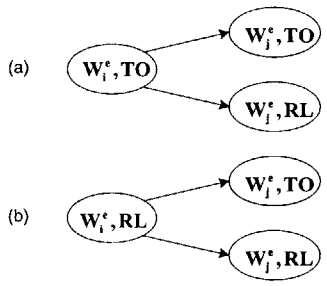


Figure 3-2: Typical Transitions in Markov Chain for System with Reno Algorithm

In order to obtain the throughput of the system, for each transition of the Markov chain, we further define the following two reward functions:

- $V_{ij}^n$: the expected number of successfully transmitted packets during the transition from state $i$ to state $j$.
- $V_{ij}^t$: the expected time taken for the transition from state $i$ to state $j$.

The corresponding reward functions associated with state $i$, denoted by $v_i^n$ and $v_i^t$, respectively, are thus

$$v_i^n = \sum_j V_{ij}^n P_{ij}^T \text{ and } v_i^t = \sum_j V_{ij}^t P_{ij}^T, \tag{3.1}$$

where $P_{ij}^T$ is the transition probability from state $i$ to state $j$. Here the superscript $T$ denotes transition. Let $\pi_i$ be the steady state distribution of the Markov chain. The steady state expected rewards per transition are therefore given by [22],

$$v^n = \sum_i \pi_i v_i^n \text{ and } v^t = \sum_i \pi_i v_i^t. \tag{3.2}$$

**Theorem 3.1.** *The throughput of the system described is* $\lambda = \frac{v^n}{v^t}$.

*Proof.* Starting from time 0, let $N(t)$ and $M(t)$ be the number of successfully transmitted packets and the number of transitions of the Markov chain up to and including time $t$, respectively. Let $N_i$ be the number of successfully transmitted packets during the $i$th transition, and $T_i$ be the time taken by the $i$th transition. Moreover, let $t_0$ be the epoch of the first transition. Then:

$$\begin{aligned} N(t_0) + \sum_{i=2}^{M(t)} N_i \quad &\leq N(t) \leq \quad N(t_0) + \sum_{i=2}^{M(t)+1} N_i \\ t_0 + \sum_{i=2}^{M(t)} T_i \quad &\leq t \leq \quad t_0 + \sum_{i=2}^{M(t)+1} T_i \end{aligned}$$

Therefore, $\frac{N(t)}{t}$ is upper bounded by $\frac{N(t_0)+\sum_{i=2}^{M(t)+1} N_i}{t_0+\sum_{i=2}^{M(t)} T_i}$ and lower bounded by $\frac{N(t_0)+\sum_{i=2}^{M(t)} N_i}{t_0+\sum_{i=2}^{M(t)+1} T_i}$. For the upper bound, we have:

$$\begin{aligned} & \lim_{t\to\infty} \frac{N(t_0) + \sum_{i=2}^{M(t)+1} N_i}{t_0 + \sum_{i=2}^{M(t)} T_i} \\ = & \lim_{M(t)\to\infty} \frac{N(t_0) + \sum_{i=2}^{M(t)+1} N_i}{t_0 + \sum_{i=2}^{M(t)} T_i} \\ = & \lim_{M(t)\to\infty} \frac{(N(t_0) + \sum_{i=2}^{M(t)+1} N_i)/M(t)}{(t_0 + \sum_{i=2}^{M(t)} T_i)/(M(t)-1)} \\ = & \lim_{M(t)\to\infty} \frac{\sum_{i=2}^{M(t)+1} N_i/M(t)}{\sum_{i=2}^{M(t)} T_i/(M(t)-1)} \\ = & \frac{v^n}{v^t}. \end{aligned}$$

Similarly, the limit of the lower bound as $t \to \infty$ can be shown to be $\frac{v^n}{v^t}$ as well. Thus, the throughput of the system is:

$$\lambda = \lim_{t\to\infty} \frac{N(t)}{t} = \frac{v^n}{v^t}. \tag{3.3}$$

□

In the next subsection, we derive the transition probabilities of the Markov chain as well as the reward functions for each transition. Then the throughput of the system can be obtained using Equation (3.1), (3.2) and (3.3).

### 3.3.2 Transition Probabilities and Reward Functions

First consider the system behavior during the transition from an arbitrary state $i$ with $(W_i^e, S_i = TO)$ to state $j$. By the definition of states and the algorithm, at the beginning of the transition, $W = 1$ and $W_t = W_i^e/2$. The window size then evolves as 1,2,4,...,$W_c$, $W_c + 1$,...,$W_M$, where $W_c$ is the least integer that is a power of two and greater than $W_t$. Denote this set of window sizes by $L_i$ and the $k$th element by $w_{ik}$. Notice that $L_i$ is a function of state $i$ only. Obviously, any possible end state $j$ must have $W_j^e \in L_i$. Let $N_{ij}$ denote the integer such that $w_{iN_{ij}} = W_j^e$. Then by the definition of the system, we know exactly how the system behaves during the transition for state $i$ to state $j$ as follows: the TL sends out windows of packets with size $w_{i1}$,$w_{i2}$,...,$w_{iN_{ij}}$. All packets in the previous $N_{ij} - 1$ windows were successfully received, and at least one packet in the last window (the $N_{ij}$th window) incurs a loss. The first loss detected within the last window is of type $S_j$.

For state $i$ with $S_i = RL$, the system behaves similarly during the transition from state $i$ to state $j$. The only difference is that now the set $L_i = \{W_t, W_t + 1, ..., W_M\}$.

The system behavior described above shows that the system can be analyzed based on windows. Specifically, since each window incurs a timeout or a random loss independently, the transition probability from state $i$ to state $j$, $P_{ij}^T$, becomes the product of probabilities of no loss on the first $N_{ij} - 1$ windows multiplied by the probability of loss $S_j$ on the last window. For the same reason, the probabilities that no loss signals are received for different windows with same size $w$ are the same. Denote it by $Q_w^W$, where the subscript $w$ is the window size and the superscript $W$ denotes that the quantity is related to a window. Similarly, the probabilities that a timeout signal or a random loss signal is received for different windows with same size $w$ are the same as well. Denote them by $P_w^{W,TO}$ and $P_w^{W,RL}$, respectively. Then, the transition probability can be expressed as:

$$P_{ij}^T = \begin{cases} 0 & \text{if } W_j^e \notin L_i \\ (\prod_{n=1}^{N_{ij}-1} Q_{w_{in}}^W) P_{w_{iN_{ij}}}^{W,TO} & \text{if } W_j^e \in L_i \text{ and } S_j = TO \\ (\prod_{n=1}^{N_{ij}-1} Q_{w_{in}}^W) P_{w_{iN_{ij}}}^{W,RL} & \text{if } W_j^e \in L_i \text{ and } S_j = RL \end{cases} \tag{3.4}$$

The reward function $V_{ij}^n$ can be obtained in a similar way as follows. From the system behavior presented previously, the number of successfully transmitted packets during the transition from state $i$ to state $j$ is the sum of the first $N_{ij} - 1$ windows of packets plus the number of successfully transmitted packets of the last window (the $N_{ij}$th window), that is,

$\sum_{n=1}^{N_{ij}-1} w_{in} + k - 1$, where $k$ is the integer such that the first $k-1$ packets in the $N_{ij}$th window were successfully received and the $k$th packet incurs the loss of type $S_j$. Note that $k$ is a random variable.

Let $P_k^{P,TO}$ and $P_k^{P,RL}$ denote the probabilities that a timeout signal and a random loss signal on the $k$th packet of a window is received, respectively. Here the superscript $P$ indicates that the quantity is related to packets. Since within the same window, packets incur losses independent of later packets, these two quantities are independent of the window size, given that the window size is larger than or equals to $k$. Then, by straightforward derivation, the expected reward $V_{ij}^n$ can be shown to have the following window-based sum form:

$$V_{ij}^n = \begin{cases} 0 & \text{if } W_j^e \notin L_i \\ \sum_{n=1}^{N_{ij}-1} w_{in} + \frac{\sum_{k=1}^{w_{iN_{ij}}} (k-1) P_k^{P,TO}}{P_{w_{iN_{ij}}}^{W,TO}} & \text{if } W_j^e \in L_i \text{ and } S_j = TO \\ \sum_{n=1}^{N_{ij}-1} w_{in} + \frac{\sum_{k=1}^{w_{iN_{ij}}} (k-1) P_k^{P,RL}}{P_{w_{iN_{ij}}}^{W,RL}} & \text{if } W_j^e \in L_i \text{ and } S_j = RL \end{cases} \tag{3.5}$$

Similarly, the reward function $V_{ij}^t$ is the sum of the expected time taken by the $N_{ij}$ windows. Let $\overline{T_w^W}$ denote the expected time taken by a window with size $w$ and no loss, and $\overline{T_w^{W,RL}}$ denote the expected time taken by a window with size $w$ and a random loss signal. Notice that the expected time taken by a window with a timeout signal is fixed to be $TO$. Then $V_{ij}^t$ has the following window-based sum form:

$$V_{ij}^t = \begin{cases} 0 & \text{if } W_j^e \notin L_i \\ \sum_{n=1}^{N_{ij}-1} \overline{T_{w_{in}}^W} + TO & \text{if } W_j^e \in L_i \text{ and } S_j = TO \\ \sum_{n=1}^{N_{ij}-1} \overline{T_{w_{in}}^W} + \overline{T_{w_{iN_{ij}}}^{W,RL}} & \text{if} W_j^e \in L_i \text{ and } S_j = RL \end{cases} \tag{3.6}$$

Equation (3.4), (3.5) and (3.6) express the transition probabilities and reward functions in terms of five unknown probabilities $Q_w^W$, $P_k^{P,TO}$ and $P_k^{P,RL}$, $P_w^{W,TO}$ and $P_w^{W,RL}$, as well as two expected time $\overline{T_w^W}$ and $\overline{T_w^{W,RL}}$. The next subsection further examines the relationship between different probabilities and reduces the number of unknown probabilities to two.

### 3.3.3 Relationship between Probabilities

As mentioned before, the possible long time taken by the retransmissions of the ARQLL ARQ due to the error-prone link transmission errors can cause the TL timeout. Since the

TL timeout signal is designed to recover the losses that cannot be recovered by the ARQLL, we call the TL timeout caused by the ARQLL retransmissions the false timeout, and let $Q_k^{P,F}$ denote the probability of no false timeout on the first $k$ packets of one window, given no random losses on the first $k$ packets. Here the superscript $P$ indicates that the quantity is related to packets, and $F$ means false timeout.

Let $Q_k^P$ be the probability that no loss signal is received on the first $k$ packets of a window with size larger than or equals to $k$, and $P_k^P$ be the probability that a loss signal on the $k$th packet of a window is received. Here again, the superscript $P$ means that the quantity is related to packets. Since each packet incurs a random loss independent of each other and independent of the ARQLL retransmissions, we have:

$$Q_k^P = Q_k^{P,F} q^k. \tag{3.7}$$

By definitions of the following probabilities and noting that the events of receiving a timeout signal and a random loss signal are exclusive, we have

$$Q_w^W = Q_w^P, \tag{3.8a}$$

$$P_k^P = Q_{k-1}^P - Q_k^P \qquad \text{with } Q_0^P = 1, \tag{3.8b}$$

$$P_k^P = P_k^{P,TO} + P_k^{P,RL} \tag{3.8c}$$

$$P_w^{W,TO} = \sum_{k=1}^{w} P_k^{P,TO} \tag{3.8d}$$

$$P_w^{W,RL} = \sum_{k=1}^{w} P_k^{P,RL}. \tag{3.8e}$$

The above equations show that once we have $Q_k^{P,F}$ and $P_k^{P,RL}$, we can obtain the five probabilities needed, $Q_w^W$, $P_k^{P,TO}$ and $P_k^{P,RL}$, $P_w^{W,TO}$ and $P_w^{W,RL}$. Specifically, Equation (3.7) gives $Q_k^P$ from $Q_k^{P,F}$. Then $Q_w^W$ can be obtained from (3.8a). Equation (3.8b) then gives $P_k^P$. Together with the given $P_k^{P,RL}$, $P_k^{P,TO}$ can be obtained from (3.8c). Finally, $P_w^{W,TO}$ and $P_w^{W,RL}$ can be obtained from (3.8d) and (3.8e). From these five probabilities, together with the quantities $\overline{T_w^W}$ and $\overline{T_w^{W,RL}}$, we can solve the transition probabilities and the reward functions. The next section derives the LL queuing models for both GBN and SRP and gives us these two probabilities, $Q_k^{P,F}$ and $P_k^{P,RL}$, and the two expected times $\overline{T_w^W}$

and $\overline{T_w^{W,RL}}$.

## 3.4 Link Layer ARQ Queuing Models

Section 3.3 shows that for the system with the Reno mechanism, the throughput of the system can be obtained once we know the four quantities $Q_k^{P,F}$, $P_k^{P,RL}$, $\overline{T_w^W}$ and $\overline{T_w^{W,RL}}$. This section develops simple queuing models for the ARQLL ARQ protocols with batch arrivals and derives these four quantities.

### 3.4.1 Queuing Model for the GBN Protocol

This subsection first considers the distribution of packet round trip times and describes the GBN queuing model. Then based on the queuing model, the four quantities required are derived.

Let $RTTH_k$ denote the TL round trip time of the $k$th packet given that the $k$th packet incurs no random loss, and $RTTL_k$ denote its ARQLL round trip time. Since in GBN the packets are acknowledged in order at ARQLL and delivered to the corresponding higher layer in order as well, we have

$$RTTH_k = RTTL_k, \tag{3.9a}$$

$$RTTH_l \leq RTTH_k \text{ for all } l \leq k. \tag{3.9b}$$

Moreover, the GBN protocol in our model can be modelled as a queuing system with independent service time (Chapter 2 Section 2.1 and [12]), denoted by $X$, with distribution $Pr(X = md + 1) = p_l^m q_l$, where $m$ is the total number of retransmissions. Since packets arrive at the ARQLL in batches and are acknowledged in order, the ARQLL round trip time of the $k$th packet $RTTL_k$ is thus the sum of its service time and the service times of the $k-1$ previous packets within the same window. Therefore, $RTTL_k$, so does $RTTH_k$, has the following distribution:

$$
\begin{aligned}
& Pr(RTTH_k = md + k + d - 1) \\
=& Pr(RTTL_k = md + k + d - 1) \\
=& \begin{pmatrix} m+k-1 \\ m \end{pmatrix} p_l^m q_l^k,
\end{aligned} \tag{3.10}
$$

where $m$ is the total number of retransmissions of the first $k$ packets in the window. Note that extra $d-1$ slots are added for allowing the GBN ACK to come back to the ARQLL sender after the packet leaves the ARQLL sender's queue.

Next we derive the four quantities based on the distribution of $RTTH_k$ given in Equation (3.10). First consider $Q_k^{P,F}$. By the definition of $Q_k^{P,F}$, together with Equation (3.9b) and the distribution of $RTTH_k$ in (3.10), $Q_k^{P,F}$ is given by

$$
\begin{aligned}
Q_k^{P,F} &\triangleq Pr(\bigcap_{l=1}^{k} RTTH_l < TO) \\
&= Pr(RTTH_k < TO) \\
&= \sum_{m=0}^{M_k^{TO}} \begin{pmatrix} m+k-1 \\ m \end{pmatrix} p_l^m q_l^k,
\end{aligned} \tag{3.11}
$$

where $M_k^{TO} \triangleq \lfloor \frac{TO-k-d}{d} \rfloor$. Notice that the physical meaning of $M_k^{TO}$ is the maximum number of retransmissions for the first $k$ packets without causing a false timeout.

Now consider $P_k^{P,RL}$. Let $F_k$ be the event that the first $k-1$ packets of a window incur no random losses but the $k$th packet incurs a random loss. Then $F_k$ is independent of $RTTH_l$ for all $l$ and $Pr(F_k) = q^{k-1}p$. The probability $P_k^{P,RL}$, which is defined before to be the probability that a random loss signal on the $k$th packet of a window is received, can thus be obtained from its definition as follows:

$$
\begin{aligned}
P_k^{P,RL} &\triangleq Pr(\bigcap_{l=1}^{k-1} RTTH_l < TO, RTTH_k + t_d \leq TO, F_k) \\
&= q^{k-1}pPr(RTTH_k \leq TO - t_d) \\
&= q^{k-1}p \sum_{m=0}^{M_k^{RL}} \begin{pmatrix} m+k-1 \\ m \end{pmatrix} p_l^m q_l^k,
\end{aligned} \tag{3.12}
$$

where $M_k^{RL} \triangleq \lfloor \frac{TO-t_d-k-d+1}{d} \rfloor$. The second equality follows from Equation (3.9b), and the

third equality follows from Equation (3.10). Notice that contrary to $M_k^{TO}$, the physical meaning of $M_k^{RL}$ is the maximum number of retransmissions for the first $k$ packets such that a random loss on the $k$th packet will be detected by a random loss signal, but not by a timeout signal.

For the same reason, i.e., the packets are delivered to the corresponding higher layer in order, the time taken by each window without loss signals is the TL round trip time of its last packet, i.e., $T_w^W = RTTH_w$. Equation (3.10) therefore gives $\overline{T_w^W}$ as follows:

$$\begin{aligned}\overline{T_w^W} &\triangleq \overline{RTTH_w|\text{no timeouts, no random losses}} \\ &= \sum_{m=0}^{M_w^{TO}} (md + w + d - 1) \begin{pmatrix} m+w-1 \\ m \end{pmatrix} p_l^m q_l^w / Q_w^{P,F}. \end{aligned} \tag{3.13}$$

Notice that $Q_w^{P,F}$ is the previous defined $Q_k^{P,F}$ with $k = w$. By the definition of $Q_k^{P,F}$, $Q_w^{P,F}$ is the probability of no timeouts and no random losses on a window with size $w$. Also notice that $M_w^{TO}$ is the previous defined $M_k^{TO}$ with $k = w$.

Similarly, $\overline{T_w^{W,RL}}$ can be obtained as follows:

$$\begin{aligned}\overline{T_w^{W,RL}} &= \sum_{k=1}^{w} \overline{RTTH_k + t_d|\text{a random loss signal on the } k\text{th packet is received}} \\ &= \sum_{k=1}^{w} q^{k-1}p \sum_{m=0}^{M_k^{RL}} (md + k + d - 1 + t_d) \begin{pmatrix} m+k-1 \\ m \end{pmatrix} p_l^m q_l^k / P_w^{W,RL} \\ &= d - 1 + t_d + \sum_{k=1}^{w} \sum_{m=0}^{M_k^{RL}} q^{k-1}p(md + k) \begin{pmatrix} m+k-1 \\ m \end{pmatrix} p_l^m q_l^k / P_w^{W,RL}. \end{aligned} \tag{3.14}$$

where in the second equality, we use Equation (3.12).

Equations (3.11) to (3.14) give us the four quantities needed for solving the system Markov chain when the ARQLL employs the GBN protocol.

### 3.4.2 Queuing Model for the SRP Protocol

As with the GBN protocol, the SRP protocol also delivers packets to the corresponding higher layer in order and Inequality (3.9b) still holds. Therefore, the first two equalities

for $Q_k^{P,F}$ and $P_k^{P,RL}$ in Equation (3.11) and (3.12) still hold. However in SRP, at the ARQLL the packets are not acknowledged in order anymore, and a packet is delivered to the corresponding higher layer only when all the previous packets and itself are correctly received by the receiver. Thus

$$RTTH_k = \max_{l=1,2,\ldots,k} RTTL_l, \tag{3.15a}$$

$$RTTH_l \leq RTTH_k \text{ for all } l \leq k. \tag{3.15b}$$

For convenience, define the function $f(k,z)$ to be the probability that the TL round trip time of the $k$th packet is less than or equal to a variable $z$, that is,

$$f(k,z) \triangleq Pr(RTTH_k \leq z). \tag{3.16}$$

Then from the first two equalities in Equations (3.11) and (3.12), $Q_k^{P,F}$ and $P_k^{P,RL}$ can be expressed in terms of $f(k,z)$ as follows

$$Q_k^{P,F} = f(k, TO-1), \tag{3.17}$$

$$P_k^{P,RL} = q^{k-1}pf(k, TO-t_d). \tag{3.18}$$

Now express $\overline{T_w^W}$ in terms of the function $f(k,z)$. Again as in the GBN protocol, in the SRP protocol the in-order delivery of packets from ARQLL to the corresponding higher layer means that $T_w^W = RTTH_w$. Moreover, $T_w^W$ is a non-negative integer valued random variable. Together with its definition, its expected value can thus be expressed as:

$$
\begin{aligned}
\overline{T_w^W} &= \sum_{z=0}^{\infty} Pr(T_w^W > z | \text{no timeout, no random loss}) \\
&= \sum_{z=0}^{\infty} Pr(RTTH_w > z | \text{no timeout, no random loss}) \\
&= \sum_{z=0}^{TO-2} (z < RTTH_w < TO | \text{no timeout}) \\
&= \sum_{z=0}^{TO-2} (1 - Pr(RTTH_w \leq z | \text{no timeout})) \\
&= TO - 1 - \sum_{z=0}^{TO-2} \frac{Pr(RTTH_w \leq z)}{Pr(\text{no timeout})} \\
&= TO - 1 - \sum_{z=0}^{TO-2} \frac{Pr(\bigcap_{i=1}^{w} RTTL_i \leq z)}{Pr(\bigcap_{i=1}^{w} RTTL_i \leq TO)} \\
&= TO - 1 - \sum_{z=w+d-1}^{TO-2} f(w,z)/f(w,TO-1). \qquad (3.19)
\end{aligned}
$$

where in the last equality, we use the fact $\min(RTTH_w) = w + d - 1$.

$\overline{T_w^{W,RL}}$ can be obtained in the similar way as that for $\overline{T_w^W}$. We omit the details and the result is:

$$
\overline{T_w^{W,RL}} = TO - \sum_{k=1}^{\min(w_T,w)} \frac{\sum_{z=k+d-1}^{TO-1-t_d} f(k,z)}{f(k,TO-t_d)} P_k^{P,RL}/P_w^{W,RL}, \qquad (3.20)
$$

where $w_T \stackrel{\triangle}{=} TO - t_d - d + 1$ is a constant.

Equation (3.17) to (3.20) show that once we know the function $f(k,z)$, the four quantities $Q_k^{PF}$, $P_k^{P,RL}$, $\overline{T_w^W}$ and $\overline{T_w^{W,RL}}$ can be obtained. For the two cases when $k \leq d$ and $k > d$, the following paragraphs derive queuing models for SRP to obtain the function $f(k,z)$. Note that by Equality (3.15a) and the definition of $f(k,z)$ (Equation (3.16)), we have

$$
f(k,z) = Pr(\bigcap_{l=1}^{k} RTTL_l \leq z). \qquad (3.21)
$$

For convenience, starting from the time the ARQLL sender receives a window of packets, we index the packets and time slots in order, that is, packet $k$ is the $k$th packet and slot $k$

is the $k$th slot.

### The Case of $k \leq d$

When the ARQLL employs the SRP protocol, the service times of packets, denoted by $X_l$ for packet $l$, are independent of each other and independent of the waiting times of the packets (Chapter 2 Section 2.1). The distribution is $Pr(X_l = (m+1)d) = p_l^m q_l$, where $m$ is the number of retransmissions before the successful transmission of packet $l$. Moreover, in the case of $k \leq d$, the waiting time of packet $l$ is fixed to be $l-1$ for all $l = 1, \ldots, k$. The ARQLL round trip times of the packets are thus $RTTL_l = l - 1 + X_l$ for $l = 1, \ldots, k$, and therefore independent of each other. Together with Equation (3.21), $f(k, z)$ can then be written in the following simple product form:

$$f(k,z) = \prod_{l=1}^{k} Pr(RTTL_l \leq z) = \prod_{l=1}^{k} (1 - p_l^{M_{zl}^r + 1}), \tag{3.22}$$

where $M_{zl}^r \triangleq \lfloor \frac{z-l+1}{d} \rfloor - 1$ is the maximum allowed total number of retransmissions of packet $l$ such that $RTTL_l \leq z$.

Equation (3.22) gives the function $f(k, z)$ when $k \leq d$.

### The Case of $k > d$

When $k > d$, packet service times are still geometrically distributed and independent of each other. However, the waiting times of packets are no longer fixed and independent of each other. As a result, $RTTL_l$ for $l = 1, \ldots, k$ are no longer independent of each other, and $f(k, z)$ no longer has the simple product form as with the case $k \leq d$ in equation (3.22). Nevertheless, we'll show that given the waiting time of packet $k$, denoted by $R_k$, to be $r$, the conditional probability $Pr(\bigcap_{l=1}^{k} RTTL_l \leq z | R_k = r)$ still has a simple product form. Bayes rule then gives a simple sum-product form of $f(k, z)$. In the following we will first derive the distribution of $R_k$, then find the conditional probability $Pr(\bigcap_{l=1}^{k} RTTL_l \leq z | R_k = r)$ and thus $f(k, z)$.

First consider the distribution of $R_k$. The event $R_k = r$ means that the first transmission of packet $k$ takes place at slot $r+1$. See Figure 3-3 for illustration. According to the SRP protocol, this happens if and only if the transmission at slot $r-d+1$ was successful and at slot $r+1$, packet $k$ is the head-of-line packet that has never been transmitted yet. The latter

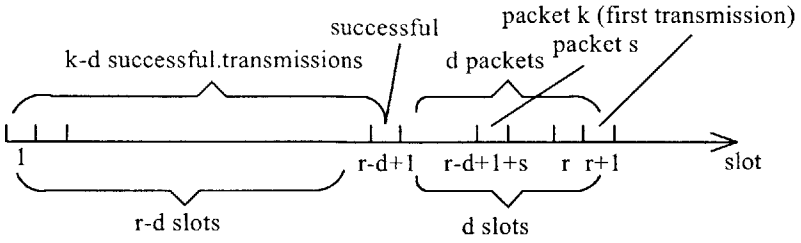


Figure 3-3: Transmission Pattern for the SRP Protocol when $k > d$

happens if and only if the number of successfully transmitted packets before and including slot $r-d+1$ is $k-d$. Since each transmission incurs a loss independently with probability $p_l$, the joint probability that this number is $k-d$ and the transmission at slot $r-d+1$ is successful is $\begin{pmatrix} r-d \\ k-d-1 \end{pmatrix} p_l^{r-k+1} q_l^{k-d}$. We therefore have

$$Pr(R_k = r) = \begin{pmatrix} r-d \\ k-d-1 \end{pmatrix} p_l^{r-k+1} q_l^{k-d}. \tag{3.23}$$

Next consider the conditional probability $Pr(\bigcap_{l=1}^{k} RTTL_l \leq z | R_k = r)$. Since the service time of packet $k$ is at least $d$, given $R_k = r$ we have $RTTL_k \geq r+1+d$. Therefore, $Pr(\bigcap_{l=1}^{k} RTTL_l \leq z) = 0$ for $z < r+1+d$. We henceforth only consider the case $z \geq r+1+d$.

Moreover, for packets that were successfully transmitted before and including slot $r-d+1$ (see Figure 3-3), their round trip times satisfy $RTTL_l \leq r-d+1+d = r+1 < z$. We hence have $f(k,z) = Pr(\bigcap_{l=1}^{k} RTTL_l \leq z | R_k = r) = Pr(\bigcap_{l \in S} RTTL_l \leq z | R_k = r)$, where $S$ is defined to be the set of packets transmitted between slot $r-d+2$ and $r+1$. Note that the size of $S$ is $d$.

For an arbitrary packet in $S$, let slot $r-d+1+s$ be the slot when it is transmitted and call it packet $s$ (see Figure 3-3 for illustration). Due to the memoryless property of geometric distribution, the remaining service time of packet $s$ is also geometrically distributed with $Pr(X_s = (m+1)d) = p_l^m q_l$, where $X_s$ denote the remaining service time and $m$ is the remaining retransmission times. Notice that $X_s$ is independent of each other for all $s = 1, ..., d$. We hence have $RTTL_s = r-d+1+s+X_s$ and given $R_k = r$, $RTTL_s$ is independent of each other for all $s = 1, ..., d$ as well. The conditional probability then becomes

$$
\begin{aligned}
& Pr(\bigcap_{l=1}^{k} RTTL_l \leq z | R_k = r) \\
= \; & Pr(\bigcap_{s \in S} RTTL_s \leq z | R_k = r) \\
= \; & \prod_{s \in S} Pr(RTTL_s \leq z | R_k = r) \\
= \; & \prod_{s=1}^{d} Pr(r - d + 1 + s + X_s \leq z) \\
= \; & \prod_{s=1}^{d} Pr(r - d + 1 + s + (m+1)d \leq z) \\
= \; & \prod_{s=1}^{d} \sum_{m=0}^{M_{zs}^r} p_l^m q_l \\
= \; & \prod_{s=1}^{d} (1 - p_l^{M_{zs}^r + 1}).
\end{aligned}
$$

where $M_{zs}^r \triangleq \lfloor \frac{z-r-s}{d} \rfloor$.

The sum-product form of the function $f(k,z)$ thus follows:

$$
\begin{aligned}
f(k,z) &= Pr(\bigcap_{l=1}^{k} RTTL_l \leq z) \\
&= \sum_{r} Pr(\bigcap_{l=1}^{k} RTTL_l \leq z | R_k = r) Pr(R_k = r) \\
&= \sum_{r=k-1}^{z-d} \prod_{s=1}^{d} (1 - p_l^{M_{zs}^r + 1}) Pr(R_k = r), \qquad (3.24)
\end{aligned}
$$

where $Pr(R_k = r)$ is given in Equation (3.23). Here in the third equality, we use the fact that the minimum waiting time for packet $k$ is $k-1$ and the maximum waiting time of packet $k$ such that its round trip time is less than or equal to $z$ is $z - d$. Equation (3.24) together with Equation (3.23) gives us the function $f(k,z)$ when $k > d$.

We conclude this section by giving the outline to numerically obtain the throughput of the system:

Given the system, protocol and loss parameters $p$, $p_l$, $d$ and $TO$:

1. For $w \in [1, W_M]$ and $k \leq w$, use the ARQLL ARQ queuing model to obtain the four quantities $Q_k^{P,F}$, $P_k^{P,RL}$, $\overline{T_w^W}$ and $\overline{T_w^{W,RL}}$ as follows:
   For GBN, plug the parameter numbers into equation (3.11) to (3.14);
   For SRP, first plug the numbers into equation (3.22) or (3.24) to get $f(k, z)$, then equation (3.17) to (3.20) give the four quantities.

2. From the two quantities $Q_k^{P,F}$ and $P_k^{P,RL}$ and relationship (3.8), solve for the five probabilities, $Q_w^W$, $P_k^{P,TO}$ and $P_k^{P,RL}$, $P_w^{W,TO}$ and $P_w^{W,RL}$.

3. From Equation (3.4) to (3.6), compute the transition probabilities and reward functions.

4. Solve the system Markov chain for the steady state distribution $\pi_i$. Then Equation (3.1) and (3.2) gives us the steady state expected rewards per transition $v^n$ and $v^t$.

5. The throughput of the system is $\lambda = \frac{v^n}{v^t}$ (Theorem 1).

## 3.5 Other Systems

### 3.5.1 System with Tahoe Algorithm

The Tahoe algorithm differs from the Reno algorithm only in that it does not use the random loss signal. All the arguments in Section 3.3 for systems with Reno algorithm are also applicable for systems with Tahoe algorithm except the parts related to random loss signals, and the model for systems with Tahoe algorithm can be obtained from the model in Section 3.3 with corresponding modifications.

Specifically, the system with the Tahoe algorithm can also be modelled as a finite state Markov chain, with the window size upon a timeout signal as the state. Unlike the chain for the system with the Reno algorithm, the chain for the system with the Tahoe algorithm has only the states with $S = TO$. The states with $S = RL$ no longer exist, and transitions from and to these states no longer have meanings. In addition, similar to those with Reno algorithm, the transition probabilities also have simple window-based product forms, and the reward functions have simple window-based sum forms as well. They can be directly obtained from their definitions in a similar way as that for the system with the Reno algorithm and expressed as follows:

$$P_{ij}^{T} = \begin{cases} 0 & \text{if } W_j^e \notin L_i \\ (\prod_{n=1}^{N_{ij}-1} Q_{w_{in}}^{W}) P_{w_{iN_{ij}}}^{W} & \text{if } W_j^e \in L_i \end{cases} \tag{3.25}$$

$$V_{ij}^{n} = \begin{cases} 0 & \text{if } W_j^e \notin L_i \\ \sum_{n=1}^{N_{ij}-1} w_{in} + \frac{\sum_{k=1}^{w_{iN_{ij}}} (k-1) P_k^P}{P_{w_{iN_{ij}}}^{W}} & \text{if } W_j^e \in L_i \end{cases} \tag{3.26}$$

$$V_{ij}^{t} = \begin{cases} 0 & \text{if } W_j^e \notin L_i \\ \sum_{n=1}^{N_{ij}-1} \overline{T_{w_{in}}^{W}} + TO & \text{if } W_j^e \in L_i \end{cases} \tag{3.27}$$

where the quantities in the above equations are defined the same as those with Reno algorithm.

Moreover, with the Tahoe algorithm, quantities related to random loss signals (those with superscription $RL$), $P_w^{W,RL}$, $P_k^{P,RL}$, and $T_w^{W,RL}$, no longer have meanings. The relationship between probabilities corresponding to equations in (3.8) can be modified by deleting these quantities, as follows:

$$Q_w^W = Q_w^P, \tag{3.28a}$$

$$P_k^P = Q_{k-1}^P - Q_k^P \qquad \text{with } Q_0^P = 1, \tag{3.28b}$$

$$P_k^P = P_k^{P,TO}, \tag{3.28c}$$

$$P_w^W = \sum_{k=1}^{w} P_k^P. \tag{3.28d}$$

Furthermore, with the Tahoe algorithm, the quantities needed from the LL queuing models to solve the transition probabilities and reward functions are reduced from four quantities to two quantities: $Q_k^{P,F}$ and $\overline{T_w^W}$. The ARQ queuing models derived in Section 3.4 are applicable for systems with Tahoe algorithm as well, and Equation (3.11) and (3.13), (3.19) and (3.17), which correspond to the GBN protocol and the SRP protocol, respectively, are still valid here. These two quantities, $Q_k^{P,F}$ and $\overline{T_w^W}$, can be obtained from these four equations for system with the GBN protocol and the SRP protocol, respectively.

The above discussions show that, the system throughput with the Tahoe algorithm can be obtained in a similar way as that with the Reno algorithm, as follows: the ARQ queuing model provides quantities $Q_k^{P,F}$ and $\overline{T_w^W}$ from Equation (3.11) and (3.13), (3.19) and (3.17),

for system with the GBN protocol and the SRP protocol, respectively. Equation (3.28) then gives all quantities needed for the transition probability $P_{ij}^{T}$ and the reward function $V_{ij}^{n}$ and $V_{ij}^{t}$. Equation (3.25), (3.26) and (3.27), together with Equation (3.1), (3.2) and Theorem 1, finally give the system throughput.

### 3.5.2 System with Multiple Transmission ARQ Protocol

When the link error probability is high, multiple transmission ARQ protocol had been proposed to improve the throughput of the link [29]. Initially the purpose of this protocol was to reduce the probability of receiver buffer overflow and thus increase the throughput of the link. Here in our system, multiple transmission attempts can reduce the effective ARQLL error probability of each packet, thus reduce the number of ARQLL retransmissions and TL false timeouts. As a results, an improved throughput is expected when the link loss probability is high. The protocol considered works as follows:

Each packet is transmitted $t$ times by the ARQLL sender. After receiving all the $t$ transmissions, if at least one of these transmissions is successful, the ARQLL receiver sends an ACK back to the sender. Otherwise the receiver sends back a NAK. The retransmission mechanism in case of NAKs works in the same way as that in the standard ARQ protocol.

Before going into the analysis of systems with the multiple transmission ARQ protocol, we first investigate the the effects of this protocol on the analysis framework defined in previous sections. For convenience, we call the set of $t$ transmissions of one packet t-transmission.

From the multiple transmission ARQ protocol, we can see that one packet incurs a link loss if and only if all its $t$ transmissions incur losses. Packet link loss probability thus reduces from $p_l$ to $(p_l)^t$.

Moreover, define the round trip delay of t-transmission to be the time between the first transmission of t-transmission and the time the sender receives an ACK/NAK corresponding to this t-transmissions. Then from the protocol we can see that the round trip delay of t-transmission is $d + t - 1$. Recall that our system is a slotted system, with one slot defined to be the transmission time of one packet. Accordingly, for systems with the multiple transmission ARQ protocol, define t-slot to be the transmission time of t-transmission. Then one t-slot is $t$ times of the original time slot. Correspondingly, the round trip delay of t-transmission measured in t-slot becomes $(d + t - 1)/t$. For simplicity, assume this number

is an integer.

With the new framework defined above, the analysis of the system with the multiple transmission ARQ protocol follows the same approach as in the previous sections. All previous formula are applicable for the system with the multiple transmission ARQ protocol with the following modifications. First, the packet link loss probability in the formula becomes $(p_l)^t$ instead of $p_l$. Second, the round trip delay becomes $(d+t-1)/t$ instead of $d$. Third, the quantities related to time, $TO$ and $t_d$, should be measured in t-slot instead of the original slot. That is, use $TO/t$ and $t_d/t$ instead of $TO$ and $t_d$ in all the formula. Finally, since the resulting reward function $v_t$ is also measured in t-slot, the resulting throughput is the number of packets per t-slot. By converting it back to the number of packets per slot, the system throughput hence is $\lambda t$ instead of $\lambda$.

## 3.6 Numerical Results and Discussions

Based on the model derived in the previous sections, here we present, for different protocol and packet loss parameters, the numerical solutions for the throughput of the system. These results tell us whether implementing ARQLL ARQ will improve the system performance, and give us some insights into the system performance as a function of the protocol and loss parameters.

In numerically solving the system, we use data that is typical in hybrid space-terrestrial networks and satellite links. As we mentioned before, this type of networks is a good example of heterogenous networks with high link loss probability and bandwidth-delay product. Usually for a satellite link, the round trip time of one transmission is around 1sec; the bit error rate BER is between $10^{-4}$ and $10^{-5}$; packet size $L$ is about 2000 bits; and transmission rate is around $10^5$ - $10^6$ bits/sec. Converting these into the parameters in our system, we obtain that the link error probability $p_l$ is between 0.02 and 0.2 and the round trip delay $d$ is about 50 - 500 time slots.

In our model, the error prone bottleneck satellite link can either employ no ARQ or one of the two ARQ protocols (GBN and SRP), and the TL can either employ the Tahoe algorithm or the Reno algorithm. Therefore, we investigate the following six systems: SRP/Reno, SRP/Tahoe, GBN/Reno, GBN/Tahoe, noARQ/Reno and noARQ/Tahoe. Specifically, the SRP/Reno system refers to the system whose ARQLL employs the SRP and TL employs

the Reno protocol. Other systems are similarly defined. Note that when the random loss probability $p = 0$ and ARQLL employs an ARQ protocol to recover link losses, there is no random loss signal generated. Consequently, the Tahoe and Reno algorithm are essentially the same. That is, in this case, the SRP/Reno system is equivalent to the SRP/Tahoe system, and the GBN/Reno system is equivalent to the GBN/Tahoe system.

### 3.6.1 Effects of Transport Layer Timeout Value

Figure 3-4 plots the throughput versus $TO$ for different systems and parameters. In particular, Figure 3-4 (a) plots the throughput versus $TO$ for all the six systems considered, with $p_l = 0.01$, $p = 0$ and $d = 100$. That is, there are no random losses and all losses can be recovered by the ARQLL ARQ. Recall that with $p = 0$, the SRP/Tahoe and SRP/Reno systems are the same, and so are the GBN/Tahoe and GBN/Reno systems. The figure shows that for systems with ARQLL ARQ (upper two curves in Figure 3-4 (a)), either GBN or SRP, the throughput increases monotonically with $TO$. This is because higher $TO$ allows more time for the ARQLL ARQ to recover link losses and reduces the false timeout probability. Whereas for systems without ARQ (lower two curves in Figure 3-4 (a)), the throughput decreases monotonically with $TO$. This is because in this case, the only way for the system to detect the losses is via timeout signals. Higher $TO$ makes the system take longer time to detect losses and recover them, which results in lower throughput. Notice that in most cases except when $TO$ is very low ($TO \leq 2d$), the systems with ARQLL ARQ have higher throughput than the systems without ARQLL ARQ by a factor of two to eight.

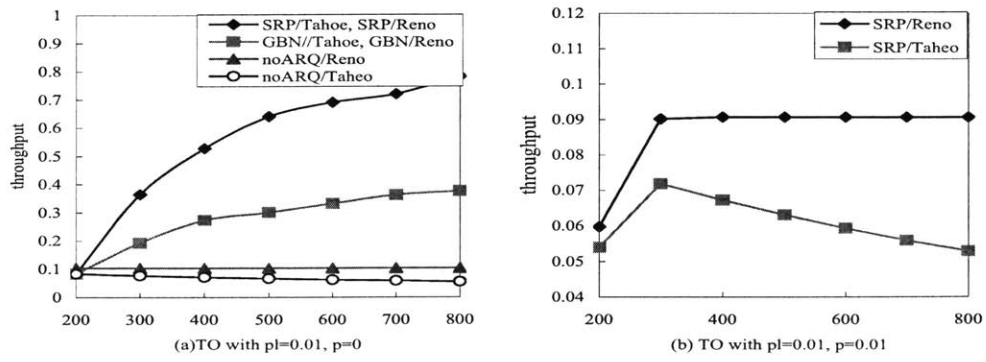


Figure 3-4: Throughput vs TL Timeout Value $TO$

Figure 3-4 (b) plots the throughput against $TO$ for the SRP/Reno and SRP/Tahoe

systems, with $p_l = 0.01$, $p = 0.01$ and $d = 100$. The curves for the GBN/Reno and GBN/Tahoe systems are similar and have the same shape. For clarity, they are not shown here. The figure shows that when both losses exist, the throughput of the SRP/Tahoe system first increases with $TO$, then decreases, while the throughput of the SRP/Reno system first increases with $TO$, then saturates. The increase part of both systems is because of the lower false timeout probability with increasing $TO$. After $TO$ reaches a certain point, further increasing $TO$ will not reduce the probability of false timeouts significantly, and the increase of the throughput for this reason should not be significant.

In addition, after $TO$ is already large, for the SRP/Tahoe system, the time for the system to detect and recover random losses becomes longer. Consequently, the reduction of the throughput due to this longer time dominates, and the throughput goes down. Together with the increasing part explained above, for the SRP/Tahoe system, the curve is concave, and there exists an optimal value of $TO$ that gives the best system performance. The optimal value depends on the link and loss parameters. On the contrary, when the $TO$ is already large, for the SRP/Reno system, the losses will more likely be detected by random loss signals rather than by timeout signals. In this sense, higher $TO$ should not affect the detection of losses, thus the throughput, significantly. Together with the increasing part when $TO$ is not large and the insignificant effect on the false timeout probability when $TO$ is large explained above, the throughput of the SRP/Reno system first increases, then saturates.

### 3.6.2 Effects of Loss Probabilities and Round Trip Delay

Figure 3-5 (a) plots the throughput against the link loss probability $p_l$ for $p = 0$, $d = 100$ and $TO = 500$. The figure shows that the throughput of the systems decreases when $p_l$ increases, regardless which TL and ARQLL protocol are used. This result is consistent with our intuition on the effects of $p_l$ on the system performance. The reason behind it is that larger loss probability means more retransmissions, longer delays and more window reductions by the TL window flow control. In addition, the figure also shows significant improvement of the system performance by implementing ARQLL ARQ: an order of magnitude at the maximum.

Our numerical results for the throughput versus the round trip time $d$ (which are not shown here) give us similar plots as those in Figure 3-5 (a). That is, the throughput of

the system decreases as $d$ increases and significant improvement of the system performance can be achieved by implementing ARQLL ARQ. This result is also consistent with the system operation, since longer round trip time for one transmission means that longer time is needed for both the TL sender and the ARQ ARQLL sender to receive the loss signals and start retransmissions, which leads to a lower throughput.

For the SRP/Reno, GBN/Reno and noARQ/Reno systems, Figure 3-5 (b) shows the throughput with respect to the random loss probability $p$ when $p_l = 0.1$, $d = 100$, $TO = 500$ and $t_d = 50$. This figure also shows improvement of the system performance by implementing ARQ. The advantage of ARQ becomes small when $p$ is large. In this case random losses come to dominate, which results in less efficiency of the ARQLL ARQ in increasing the throughput. Our results for the SRP/Tahoe, GBN/Tahoe and noARQ/Tahoe systems yield similar observations. For brevity, these plots are not shown here.

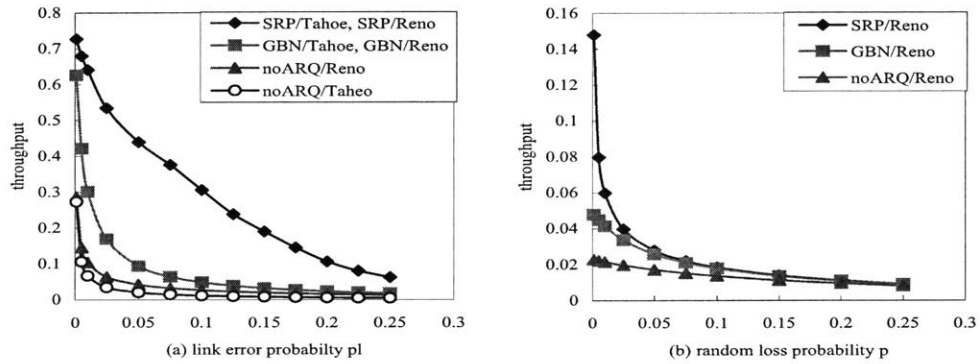


Figure 3-5: Throughput vs Error Probability

### 3.6.3 Effects of Other Parameters

Figure 3-6 (a) shows the throughput as a function of $t$, the number of transmission attempts per packet at ARQLL, when $p_l = 0.2$, $p = 0.01$, $d = 100$, $TO = 500$. It shows that for each system, the throughput first increases then decreases. Actually there are two opposite effects by increasing $t$. On the one hand, compared with standard ARQ protocol where $t = 1$, transmitting each packet $t$ times with $t > 1$ greatly reduces the effective link loss probability seen by the ARQLL by a power of $t$, which results in significant reduction of the number of retransmissions at the ARQLL. Because of the large delay of the error-prone link, this reduction of the retransmission times not only save a significant amount of time

for one packet, but also leads to significant decrease of the variability of the packet round trip times, and thus the number of the TL false timeouts and the TL window reductions. In this sense, higher $t$ may lead to a higher throughput. On the other hand, transmitting each packet $t$ times also means that the effective link capacity is reduced by a factor of $t$, therefore greatly reduces the throughput. As a result, the overall system performance against $t$ depends on the tradeoff between these two effects. Figure 3-6 (a) shows that when $t$ is relatively small, the benefit of increasing $t$ dominates, while when $t$ is relatively big, the drawback dominates. The curve is thus concave, and there exist an optimal value of $t$ which gives the best throughput. Again, the optimal value depends on the link and protocol parameters. Notice that the figure shows a significant increase in throughput by using the optimum $t$ (for example, a factor of 4 for the GBN/Reno system).

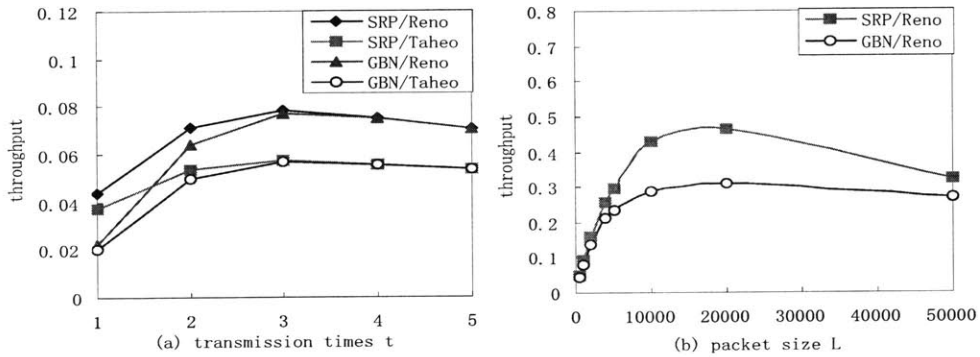


Figure 3-6: Throughput vs ARQLL Transmission Times and Packet Size

Figure 3-6 (b) plots the throughput as a function of the packet size $L$ for the SRP/Reno and GBN/Reno systems, when the random loss probability $p = 0.01$, the round trip delay of the link is 1sec, BER= $10^{-5}$ and the transmission rate is $10^5$ bits/sec. Here in order to illustrate the effects, we deliberately extend the range of packet lengths and make the length go beyond the practical values. The figure shows that the throughput first goes up then goes down with increasing $L$. The increase in throughput is because of the decrease in ARQLL round trip delay $d$ (as measured in time slots) due to increasing $L$, and the decrease in throughput is due to the increase in packet error probability $p_l$ (again, because of the larger packet size). There exists an optimal value of $L$ that gives the best performance. The optimal value depends on the link parameters and the protocols employed, and can significantly improve the throughput as well (a factor of 6 to 9 with the parameters dis-

cussed). Our numerical results for the other four systems, the SRP/Tahoe and GBN/Tahoe systems and the noARQ/Reno and noARQ/Reno systems, give the same trends as those in Figure 3-6 (b). For brevity, they are not shown here.

### 3.6.4 Is ARQ Beneficial?

Now let us assess whether the ARQLL ARQ is beneficial or not. First consider the systems with the Tahoe protocol. In these systems, implementing the ARQLL ARQ can recover some of the link losses (which will otherwise lead to TL timeouts) and thus decrease the number of window reductions by the TL. Therefore the ARQLL should improve the throughput and be beneficial. This is evidenced by Figure 3-4 to Figure 3-6, where the SRP/Tahoe and GBN/Tahoe systems have better performance than the noARQ/Tahoe system .

Whereas when the TL employs the Reno protocol, the ARQLL has two opposite effects on the system performance. On one hand, the ARQLL can recover some of the link losses, and thus decrease the number of window reductions, which would lead the system back to the congestion avoidance phase if no ARQ were employed. As a result, the throughput should increase. On the other hand, the ARQLL introduces false timeouts. These false timeouts lead the system back to the slow start phase instead of the congestion avoidance phase if no ARQ were employed. Consequently, the throughput should decrease. Figure 3-4 to Figure 3-6 show that, in all cases except when the TL timeout value $TO$ is very small ($TO \leq 2d$, see Figure 3-4 (a)), the benefit of employing ARQ overcomes the drawback and the SRP/Reno and GBN/Reno systems give better performance than the noARQ/Reno system.

Since in most cases except when $TO$ is very low, employing ARQ yields significant improvement of the system throughput (for example, a factor of two to eight from Figure 3-4 (a) and an order of magnitude at the maximum in Figure 3-5 (a)), we thus suggest the implementation of ARQ over the satellite link.

## 3.7 Conclusions

In this chapter we provide an exact model for heterogenous data networks with AIMD transport protocols and link layer ARQ protocols. Both GBN and SRP are considered, and the delay of ARQ ACKs is also taken into account. Numerical solutions for the throughput

as a function of different protocol and packet loss parameters are presented. Our analysis shows that for most situations of interest, it is better for the error-prone link to implement ARQ.

We were also able to obtain some insights on the system performance as a function of various protocol parameters, such as the transport layer time-out value; the link layer packet size and the number of transmission attempts per packet at the link layer. We show that by proper setting of these parameters, system throughput can be improved by nearly an order of magnitude.

There are two natural extensions to the work presented in this chapter. One is to analyze TCP performance when there exist multiple connections. The other is to include non-persistent users, that is, users that have limited packets to sent.

# Chapter 4

# Interactions between TCP and Random Access Scheme

In this chapter we will investigate the interaction between TCP and ALOHA by analyzing a hybrid terrestrial-satellite network with TCP at the transport layer and ALOHA at the MAC layer. The system performance will be derived analytically, and the impacts of protocols and parameters will be studied in detail. Simulations will also be presented which confirms the analysis.

## 4.1 Introduction

Often in satellite networks, a group of ground terminals share the same channel to one satellite for transmissions. Compared to other multiple access protocols, such as Frequency Division Multiple Access (FDMA) and Time Division Multiple Access (TDMA), random access protocols, such as ALOHA, have the advantages of simple implementation and good performance over bursty and sporadic traffic ([12]). Moreover, with random access protocols, it is easy to add and remove users. Hence, random access protocols are widely employed in satellite and wireless networks. In addition, currently TCP is the dominate transport layer protocol that provides reliable end-to-end transmissions and is widely implemented in data networks. It is thus of interest to study the system performance with TCP at the transport layer and ALOHA at the MAC layer.

Since ALOHA was first proposed in 1970s, there have been many papers analyzing its performance. In [12] and the papers therein, the authors give comprehensive and rigorous

analysis on several aspects of ALOHA systems, such as the throughput, collision probability and idle probability, and stability issues. These analysis considers ALOHA protocols only and disregards the effects of other layers. For idealized slotted ALOHA, one of the main assumptions these analysis make is that packets arrive at senders according to independent Poisson processes. Two of the main results on its performance are that, the maximum system throughput is $1/e$, and this throughput can be reached by adjusting the transmission probability to make the system attempt rate be one. At other attempt rate, the system has either too many idle slots or too many collisions, and the throughput is below $1/e$.

In practical networks, MAC layers receives packets from their corresponding higher layers. When a system employs TCP at its transport layer, the packets arrival process at the MAC layer is no longer a Poisson process but under the control of TCP window flow control scheme. Moreover, the maximum number of packets available at the MAC layer is limited by the current TCP window size.

In addition, from the perspective of TCP, there have been many papers studying the TCP performance in hybrid terrestrial-satellite networks [4, 50, 49, 45, 7, 27, 3, 10, 26, 40, 41]. Most of these work focuses on the properties of high bandwidth-delay product and high link loss property. The paper [43] and [36] also analyze the performance of a normal TCP session with random packet loss probability. This chapter also considers the TCP performance in hybrid terrestrial-satellite networks, but focuses on its interaction with the MAC layer random access scheme.

In particular, under random access protocols, packets enter the shared channel randomly, and collisions occur at the MAC layer with some probability. These MAC layer collisions directly affect TCP window evolution at the transport layer and hence the TCP performance. In addition, ideal ALOHA protocol resolves collisions by MAC layer retransmissions of collided packets. When the system employs TCP at the transport layer, there is one more option, that is, resolving collisions by TCP retransmissions.

In this chapter we study these interactions between TCP and ALOHA by examining the system performance with TCP at the transport layer and ALOHA at the MAC layer. Specifically, the system investigated is a hybrid terrestrial-satellite network with a large number of identical persistent source-destination pairs. Each pair employs TCP at its transport layer and each source employs the slotted ALOHA random access scheme at its MAC layer.

An analytical model for the system considered is developed, and two simple equations are derived from which the system performance can be solved directly. The maximum possible system throughput is shown to be $1/e$, and a sufficient and necessary condition to achieve this throughput is given. Moreover, the optimal MAC layer transmission probability at which the system achieves its highest throughput is also derived. The impacts of different protocols and parameters on the system performance, such as MAC layer transmission probability, TCP timeout backoff and MAC layer retransmissions, are studied in detail.

The rest of the chapter is organized as follows: in the next section we give a detail description on the systems and protocols examined. Then in Section 4.3 and 4.4, we analyze the system with and without TCP timeout backoff, respectively, and derive formulas for the system performance and conditions for achieving the maximum throughput, as well as the optimal MAC layer transmission probability. In Section 4.5 we discuss the impact of TCP timeout backoff on the system considered, and evaluates MAC layer retransmissions. Finally, in Section 4.6 we conclude the chapter.

## 4.2 System Description

The system we consider consists of $N$ identical source-destination pairs (SD pairs), where $N$ is large, as shown in Figure 4-1. All $N$ sources share a common channel to the satellite. Each source has unlimited number of packets to be sent to its corresponding destination. Each SD pair employs TCP at its transport layer for congestion control purpose, where a selective repeat retransmission scheme is employed and only those packets that are lost will be retransmitted. At the destinations packets are delivered from lower layers to TCP in order. All sources employ an ALOHA multi-access scheme at their MAC layer for multi-access purpose. The two protocols considered, TCP and ALOHA, will be described in detail later.

For all SD pairs, all packets at the MAC layer have the same length and each packet requires one time unit, called a slot, for transmission. The round trip time between each SD pair, defined to be the duration between the time a packet is sent out by the MAC layer and the time the source receives its acknowledgement, is a random variable with mean $D$ slots. Here the randomness represents the queuing delays and other random delays experienced by the packets. To focus on the impact of random access, we assume that there are no other

packet losses in the network except losses due to MAC layer collisions.

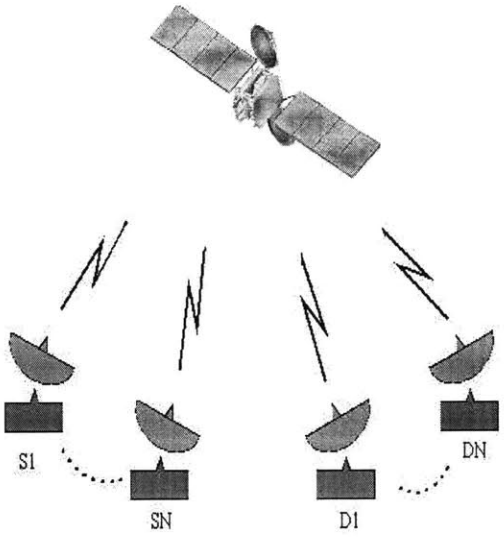


Figure 4-1: System with TCP over MAC Random Access

The transport layer window evolution follows the rules of a simplified TCP Reno window evolution. Specifically, the simplified TCP Reno keeps two parameters: current window size $W$ and window threshold $W_t$. Its window flow control algorithm works as follows:

- The initial window size is 1.
- Upon an acknowledgment:
  - If $W < W_t$, then $W \leftarrow W + 1$;
  - If $W \geq W_t$, then $W \leftarrow W + 1/\lfloor W \rfloor$.
- Upon a timeout signal, $W_t \leftarrow W/2$ and $W \leftarrow 1$.

  Upon three duplicate ACKs, $W_t \leftarrow W/2$ and $W \leftarrow W_t$.

The phase with $W < W_t$ is called slow start phase, and the phase with $W \geq W_t$ is called congestion avoidance phase. Notice that since what we are interested in are TCP's retransmissions and congestion control, here we deliberately disregard other TCP issues, such as connection establish and termination, and time granularity.

The ALOHA multi-access scheme considered works as follows:

- At each source, when one or more packets are available in the MAC layer buffer, its MAC layer transmits the first packet with probability $p$.
- If two or more sources transmit packets in a given slot, a collision occurs and all packets transmitted are assumed lost.
- If only one source transmits a packet in a given slot, the packet will be successfully received.

Note that this scheme does not allow MAC layer retransmissions of collided packets. TCP retransmission hence is the only way to recover collided packets. The system with MAC layer retransmissions will be discussed later in Section 4.5. Also note that according to the scheme, each source MAC layer can be viewed as a queuing system, with all arrivals coming from its corresponding transport layer and geometrically distributed service time with mean $1/p$.

Our objective is to analyze the system described above, that is, with TCP at the transport layer and ALOHA at the MAC layer. For convenience, we call the system the TCP system. However, due to its complex window update algorithm, it is usually complicated to analyze TCP. Nevertheless, our system employs ALOHA at the MAC layer, and collisions occur with high probability. Hence, the TCP window size is expected to be small with high probability. We therefore propose a system whose window update algorithm consists of only the congestion avoidance phase, and call the system the CA (Congestion Avoidance) system. Specifically, the window of the CA system is updated according to the following rules:

- Initially the window size $W = 1$.
- Upon an acknowledgement, the window size is updated to be $W \leftarrow W + \frac{1}{[W]}$.
- Upon a timeout, the window size is reset to be $W \leftarrow 1$.

By comparing this WFC scheme with the TCP WFC scheme, we see that this WFC scheme is similar to that in the congestion avoidance phase of TCP but without fast retransmit/recovery. All the analysis henceforth is based on this CA system, and we therefore call the CA system the system when there is no ambiguity. As we will see later, due to the high

collision probability, the window size of the TCP system is small. Therefore, with high probability, not enough duplicate ACKs will be generated to trigger fast retransmit/recovery mechanism [21]. Moreover, the small window size of the TCP system also leads to a small window threshold, which further results in negligible slow start phase. The CA system is thus expected to be a good approximation of the TCP system, which is evidenced by the comparison of simulation results of these two systems presented in Section 4.3.3.

Note that from its window update algorithm, after each timeout, the window of the CA system has exactly the same evolution. This property allows us to model the system as a renewal process, thus greatly simplifies the analysis of the system.

We further assume that the TCP timeout value, denoted by $RTO$, is large enough that the probability of timeout due to queuing delay at the MAC layer can be ignored. Recall that there are no losses other than collision losses. Hence, TCP timeouts can only be triggered by MAC layer collisions.

In addition, to investigate the impacts of TCP timeout backoff, we also consider a TCP variation that does not employ timeout backoff. Not only the analysis of this system is the basis of the analysis of the TCP system, but also we will see later that TCP timeout backoff actually hurt the system performance.

For convenience, we borrow the concept of round from [44] at the transport layer, with extension that includes the timeout signal: a round starts with transmission of $W$ packets, where $W$ is the current window size of the congestion window. Once all packets falling within the congestion window have been sent, no other packets are sent until the first ACK for one of these $W$ packets or a timeout signal is received . This ACK reception or timeout marks the end of the current round and the beginning of the next round.

Further define a timeout interval to be an interval between two successive timeouts, and index the packets sent and the rounds within one timeout interval in order, that is, packet $k$ is the $k$th packet and round $k$ is the $k$th round. The ACK for packet $k$ is called ACK $k$. Table 4.1 illustrates the window evolution and rounds within one timeout interval for the CA system. For example, as shown in column 1 to column 3 of Table 4.1, each time when the window size is increased by 1, there are two packets released. Otherwise, there is one packet released. After the window size reaches 3, there are three ACKs received, namely ACK 3,4 and 5, which lead to a window increase to 4. Accordingly, four packets are released, namely packet 5, 6, 7 and 8. In addition, as shown in the 4th and 5th columns,

Table 4.1: Window Evolution and Rounds

| ACK | window size | packet released | round | first packet in round | number of packets in round |
|---|---|---|---|---|---|
| | 1 | 1 | 1 | 1 | 1 |
| 1 | 2 | 2,3 | 2 | 2 | 2 |
| 2 | 2 | 4 | 3 | 4 | 3 |
| 3 | 3 | 5,6 | | | |
| 4 | 3 | 7 | 4 | 7 | 4 |
| 5 | 3 | 8 | | | |
| 6 | 4 | 9,10 | | | |
| 7 | 4 | 11 | 5 | 11 | 5 |
| 8 | 4 | 12 | | | |

round 4 begins with ACK 4, which is the first ACK of packets in round 3, and the first packet in round 4 is packet 7. ACK 7 thus becomes the first ACK of packets in round 4 and marks the end of round 4 and beginning of round 5. By counting the number of packets in round 4, we obtain 4, which is shown in the last column.

## 4.3 Systems without TCP Timeout Backoff

This section analyzes CA systems without TCP timeout backoff. We will first consider the entire system including all SD pairs, and then examine one SD pair in isolation. The combined analysis yields formulas that together can be used to solve for the system performance.

### 4.3.1 System Level Analysis

For every sender/receiver pair, define the throughput as the number of packets correctly received by the receiver per unit time, and the send rate as the number of packets sent by the sender per unit time. That is, the send rate includes both packets that are included in throughput and the packets that are lost due to collisions. By assuming equivalence between time average and ensemble average, the throughput is also the probability that in a slot, a packet is correctly received by the receiver. Similarly, the send rate is also the probability that in a slot, a packet is sent by the sender.

In our system, we have two layers, the transport layer and the MAC layer; hence we have definitions of throughput and send rate at both layers. However, the MAC layers receive packets only from their corresponding transport layers. Therefore, the send rate at

both layers are the same. Moreover, collision losses at the MAC layers are the only losses in the system. So the throughput at both layers are the same as well. Henceforth, we use the term throughput and send rate to refer to the throughput and send rate at both layers. In addition, the throughput and send rate can refer to those of a single SD pairs and those of all $N$ SD pairs. We call those for a single SD pair individual throughput and send rate, and denote them by $\lambda_d$ and $B_d$, respectively. We call those for all $N$ SD pairs system throughput and send rate, and denote them by $\lambda_s$ and $B_s$, respectively.

From the above definitions and discussions of throughput and send rate, we conclude that in our system, the system throughput $\lambda_s$ is the probability that there is exactly one MAC layer sending a packet in a slot. For one particular SD pair, the individual send rate $B_d$ is the probability that the MAC layer of this SD pair sends a packet in a slot. Since we have a large number of SD pairs ($N$ is large), we further assume that for a particular SD pair, its state is independent of the state of other SD pairs. We hence have the following relationship between the system throughput, idle probability, collision probability and the individual send rate:

$$\lambda_s = NB_d(1 - B_d)^{N-1} \approx NB_d e^{-NB_d}, \tag{4.1a}$$

$$P_I = (1 - B_d)^N \approx e^{-NB_d}, \tag{4.1b}$$

$$P_C = 1 - \lambda_s - P_I. \tag{4.1c}$$

where $P_I$ and $P_C$ are the idle probability and collision probability, respectively, and the approximations hold when $B_d$ is small and $N$ is large.

Due to the independence assumption, the above analysis is similar to that of a standard ALOHA system [12]. Specifically, the equations in (4.1) show that, as in an ALOHA system, the number of packets correctly received in a slot can be approximated by a Poisson random variable, with the attempt rate $NB_d$ as its mean. The maximum possible throughput can be achieved at $NB_d = 1$, with the corresponding throughput, idle probability and collision probability being $1/e$, $1/e$ and $1 - 2/e$, respectively. However, unlike a standard ALOHA system, in our system $NB_d = 1$ is not always achievable due to the impacts of the transport layer window flow control, which will be discussed in detail later, while in a standard ALOHA system, an attempt rate of one can always be achieved with proper parameter

settings.

Furthermore, under the independence assumption, the effects of other SD pairs on one particular SD pair are aggregated into one parameter $Q$, defined to be the probability that all other $N-1$ MAC layers transmit no packets in a slot. Since all SD pairs are identical, we have:

$$Q = (1 - B_d)^{N-1} \approx e^{-(N-1)B_d} \approx e^{-NB_d}. \tag{4.2}$$

Note that although the values of $P_I$ and $Q$ are approximately the same (Equation (4.1b) and (4.2)), they have different physical meaning. $P_I$ is for all $N$ SD pairs, while $Q$ excludes the SD pair considered and is for the other $N-1$ SD pairs only.

Moreover, the independence assumption also gives $\lambda_s = N\lambda_d$ and $B_s = NB_d$.

### 4.3.2 Session Level Analysis

As mentioned before, the effects of the other $N-1$ SD pairs on one particular SD pair are aggregated into one parameter $Q$, the idle probability of all other $N-1$ SD pairs. This particular SD pair can thus be modelled as a transport layer session with collision probability of each packet being $1-Q$. However, unlike the TCP sessions with random packet losses analyzed in literatures [43, 36], in our system all TCP packets must pass the MAC layer queuing system, which makes the analysis of the TCP system much more complicated. Therefore, instead of analyzing the TCP system directly, we analyze the CA system proposed. Specifically, in this section, we use the property of the CA system to model one particular SD pair as a renewal process, and obtain an upper bound and a lower bound for its send rate, $B_d$, in terms of $Q$.

For one particular SD pair, after a timeout, consider packet transmissions before the first collision occurs. All packets sent before the first collided packet won't encounter a timeout, since they encounter no losses and the large timeout value assumption ensures that their ACKs will be received before the TCP retransmission timer expires. Whereas the first collided packet will eventually encounter a timeout, since there is no mechanism other than the transport layer retransmission to recover this collision loss. Therefore, the first collision is the one that leads to the next timeout.

Moreover, since packet transmissions between a timeout and the first collision afterwards

were successful, the transport layer sender will receive some ACKs after the collision and release some new packets. The large timeout value assumption ensures that the MAC layer will finish the transmissions of these later released packets before the next timeout. Thus upon each timeout signal, there is no packet in the MAC buffer. In addition, according to the WFC scheme, the window evolves exactly the same after each timeout signal. The timeout signal sequence therefore forms a renewal process, with a renewal cycle being a timeout interval between two successive timeouts. Let $M$ be the number of packets sent during a cycle and $T$ be the cycle length. Then by the renewal theory,

$$B_d = \frac{E[M]}{E[T]}. \tag{4.3}$$

To solve for $B_d$, first consider $E[T]$. Instead of deriving its exact expression, which requires complicate queuing analysis, we derive an upper bound and a lower bound. Let $R$ be the number of successful rounds during a cycle and $RTO$ be the timeout value that ends the cycle. Then $T$ is at least the sum of the duration of these $R$ rounds plus $RTO$. Each round takes at least the service time of its first packet, denoted by $F_k$ for round $k$, plus the two way propagation delay this packet experiences, denoted by $D_k$, for the sender to receive its ACK. We therefore have a lower bound for $T$, denoted by $T^L$, to be $T \geq T^L = \sum_{k=1}^{R}(F_k + D_k) + RTO$.

Similarly, an upper bound for $T$, denoted by $T^U$, can be obtained when ignoring the overlap between the service time of each packet and $D_k$ of each round $k$. Mathematically, let $X_k$ be the service time of packet $k$. Then $T \leq T^U = \sum_{k=1}^{M} X_k + \sum_{k=1}^{R} D_k + RTO$. Along with the lower bound $T_L$, we have

$$\sum_{k=1}^{R}(F_k + D_k) + RTO = T^L \leq T \leq T^U = \sum_{k=1}^{M} X_k + \sum_{k=1}^{R} D_k + RTO.$$

Here and henceforth, for convenience and clarity, the expectation of $RTO$ is denoted by $RTO$ as well. Note that TCP updates $RTO$ based on the measurements of round trip times seen by its packets, which takes into account both their average and standard deviation [52].

By our multi-access scheme, the service time for each packet is geometrically distributed with mean $1/p$ and independent of each other and $R$ and $M$ as well. Thus by taking expectations of the above inequality,

$$(\frac{1}{p} + D)E[R] + RTO = E[T^L] \leq E[T] \leq E[T^U] = \frac{E[M]}{p} + DE[R] + RTO. \quad (4.4)$$

Now consider $E[M]$. Define $Z$ to be the index of the packet that incurs the first collision. Then the first $Z-1$ packets were successfully transmitted and have been or will be ACKed, while packet $Z$ and packets sent thereafter won't be ACKed. According to the WFC scheme, new packets can be released only upon reception of ACKs. Therefore, $M$ equals to 1, counting for the first packet, plus the number of packets triggered by the $Z-1$ ACKs. In addition, in the WFC scheme, each ACK triggers the release of one packet, except those ACKs that increase the window size by 1 (i.e., the last ACK in a round) where two packets are released (See Table 4.1 for illustration). Let $I$ be the number of such ACKs out of the $Z-1$ ACKs, then $M = 1 + (Z-1) + I = Z + I$. Moreover, the window size is increased by 1 per round. Therefore $I = R$, and

$$E[M] = E[Z] + E[R]. \quad (4.5)$$

We now have bounds for $E[T]$ as in Inequality (4.4) and $E[M]$ as in Equation (4.5) in terms of $E[Z]$ and $E[R]$. Recall that $Q$ is the probability that no other senders send their packets in one slot and this event is independent of the state of the particular SD pair under consideration. Therefore, each packet of the particular SD pair incurs a collision with probability of $1-Q$ and independent of each other. $Z$ is thus geometrically distributed with $Pr(Z = z) = Q^{z-1}(1-Q)$ and $E[Z] = 1/(1-Q)$. By exploring the relationship between $R$ and $Z$, it can be shown that $E[R] = \sum_{k=1}^{\infty} Q^{k(k+1)/2}$. For brevity, here we omit the details. We hence have:

$$E[M] = \frac{1}{1-Q} + \sum_{k=1}^{\infty} Q^{k(k+1)/2}, \quad (4.6)$$

$$(\frac{1}{p} + D)\sum_{k=1}^{\infty} Q^{k(k+1)/2} + RTO \leq E[T] \leq \frac{1}{p(1-Q)} + (\frac{1}{p} + D)\sum_{k=1}^{\infty} Q^{k(k+1)/2} + RTO. \quad (4.7)$$

By plugging them into Equation (4.3), we obtain the following bounds for the send rate $B_d$ in terms of $Q$:

$$\frac{\frac{1}{1-Q}+\sum_{k=1}^{\infty}Q^{k(k+1)/2}}{\frac{1}{p(1-Q)}+(\frac{1}{p}+D)\sum_{k=1}^{\infty}Q^{k(k+1)/2}+RTO}=f^L\leq B_d$$
$$\leq f^U=\frac{\frac{1}{1-Q}+\sum_{k=1}^{\infty}Q^{k(k+1)/2}}{(\frac{1}{p}+D)\sum_{k=1}^{\infty}Q^{k(k+1)/2}+RTO}. \tag{4.8}$$

Notice that for large enough timeout value $RTO$, both bounds increase monotonically with $Q$. Mathematically, this can be shown by taking the derivative of the bounds with respect to $Q$. Physically, from our derivation, the lower bound $f^L$ corresponds to the send rate of the following system: after sending the first packet of each round, the MAC layer holds the transmission of other packets until it receives the ACK of the first packet. This is how we obtain the upper bound $T^U$ for the duration $T$ between two successive timeouts. Clearly, the send rate of this system increases with the idle probability of the other $N-1$ SD pairs $Q$. Similarly, the upper bound $f^U$ corresponds to the send rate of the following system: after sending the first packet of each round, the MAC layer finishes the transmission of all other outstanding packets before it receives the ACK of the first packet, that is, within time $D_k$ for round $k$. This is how we obtain the lower bound $T^L$ for the duration $T$ between two successive timeouts. Again, the send rate of this system increases with the idle probability of the other $N-1$ SD pairs $Q$.

To fully under the system behavior, the expectation of the collision window $W_z$, defined to the window size when the collided packet $Z$ is released (i.e., when the collision occurs), can also be shown to be:

$$E[W_Z]=1+\sum_{k=2}^{\infty}Q^{\frac{k(k+1)}{2}-2}. \tag{4.9}$$

The details are omitted for brevity. Note that according to the WFC scheme, the difference between the collision window size and the window size upon a timeout indication is no more than one (see Table 4.1).

### 4.3.3 System Performance

Section 4.3.1 analyzes $N$ SD pairs together and gives one relationship between the individual send rate $B_d$ and the idle probability of $N-1$ SD pairs $Q$, in Equation (4.2). Section 4.3.2 gives an upper bound and a lower bound of $B_d$ in terms of $Q$ in Inequality (4.8). Based on

these results, this section first gives bounds for $B_d$ and $Q$ in terms of system and protocol parameters, and then discusses the system performance. A sufficient and necessary condition for the system to achieve the maximum possible throughput $1/e$ is also given, as well as the optimal MAC layer transmission probability at which the throughput is maximized. The analysis is confirmed by both numerical results for the CA system and simulation results for the TCP system under various system and protocol parameters.

Before proceeding, we first introduce the settings for the system and protocol parameters in the numerical computations and simulations. For the systems, we consider the following data that is typical for a satellite link: the two-way propagation delay between each SD pair is around 1sec; packet size $L$ is about 10000 bits; and transmission rate is between 100k - 1M bps. Converting these into the parameters in our system, we obtain that the two-way propagation delay is about 10 - 100 time slots. We therefore set the round trip time of packets $D$ to be in the range of 10-100 slots.

Simulations presented are performed on the ns2 simulator [1]. Since this section considers systems without timeout backoff and Karn's Algorithm, during the simulation, we block TCP timeout backoff. Moreover, to verify the relationship derived between system performance and other system parameters, the value of $RTO$ used in the numerical computations is the average timeout value from the simulations.

For both the CA system and the TCP system, Figure 4-2 plots the simulated system throughput as a function of MAC layer transmission probability, with $D = 50$ and N=10. From the figure we can see that the throughput of both systems are very close to each other. The collision window size, which is not shown here, is below 3. Simulations for other sets of parameters show similar results, that is, the performance of the CA system and the TCP system is very close, and the collision window size is small (below 8). This justifies our conjecture that the TCP system has small window size with high probability and the CA system is a good approximation of the TCP system. All simulations presented later are hence for the TCP system unless specified otherwise.

### System Performance and Condition for Maximum Throughput

Based on the Equation (4.2) and Inequality (4.8), this subsection shows how to obtain the system performance given the system and protocol parameters.

Figure 4-3 plots the relationship between $B_d$ and $Q$ as given in Equation (4.2), called

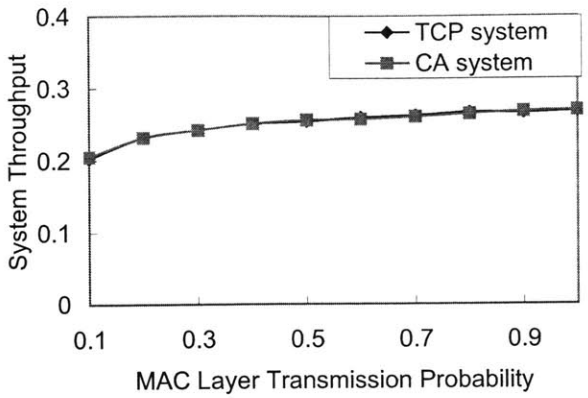


Figure 4-2: Comparison between CA System and TCP System - without Backoff ($N = 10$ and $D = 50$)

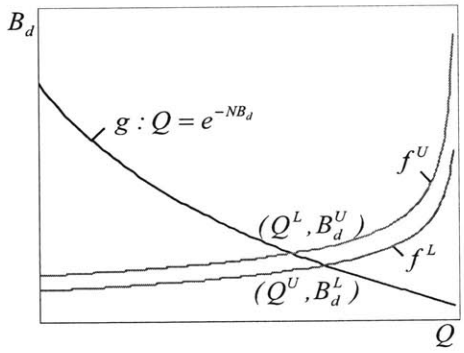


Figure 4-3: Bounds for $B_d$ and $Q$

curve $g$. Figure 4-3 also plots the bounds of $B_d$, $f^L$ and $f^U$, as given in Inequality (4.8). For clarity, the distances between the curves are exaggerated. The actual $B_d$ and $Q$ should be on curve $g$, as well as be inside the area between curve $f^L$ and $f^U$. Therefore, $B_d$ and $Q$ are on the section of curve $g$ between curve $f^L$ and $f^U$. The intersections of curve $g$ with curve $f^L$ and $f^U$ thus give a lower bound and an upper bound for $B_d$ as well as a lower bound and an upper bound for $Q$. Denote them by $B_d^L$ and $B_d^U$, $Q^L$ and $Q^U$, respectively. Therefore, $B_d^L$ and $B_d^U$ are solutions of the following two equations, respectively:

$$B_d[\frac{1}{p(1-e^{-NB_d})} + (\frac{1}{p}+D)\sum_{k=1}^{\infty} e^{-NB_d k(k+1)/2} + RTO] = \frac{1}{1-e^{-NB_d}} + \sum_{k=1}^{\infty} e^{-NB_d k(k+1)/2}, \tag{4.10a}$$

$$B_d[(\frac{1}{p}+D)\sum_{k=1}^{\infty} e^{-NB_d k(k+1)/2} + RTO] = \frac{1}{1-e^{-NB_d}} + \sum_{k=1}^{\infty} e^{-NB_d k(k+1)/2}. \tag{4.10b}$$

Figure 4-4 plots the system performance, the system throughput, idle probability and collision probability, as a function of $B_d$, given in Equation (4.1). The figure shows that the bounds for $B_d$ actually give us the range of the system performance. For example in this figure, both bounds for $B_d$, $B_d^L$ and $B_d^U$, lie within the same monotonic region of the throughput, i.e., within $[0, 1/N]$. Hence their corresponding throughput, $\lambda_s^L$ and $\lambda_s^U$ in Figure 4-4, are also lower and upper bounds for the actual throughput $\lambda_s$. Moreover, since $P_C$ and $P_I$ are monotonic with $B_d$, the collision probability and idle probability corresponding to $B_d^L$ and $B_d^U$ are also bounds for $P_C$ and $P_I$. As in Figure 4-4, $P_C^L \leq P_C \leq P_C^U$ and $P_I^L \leq P_I \leq P_I^U$. We thus conclude that the two bounds given in Equations (4.10), together with Equation (4.1), fully characterize the system performance. Note that in Figure 4-4, the range with monotonic increase throughput corresponds to systems with too many idle slots to achieve throughput of $1/e$, and the range with monotonic decrease throughput corresponds to systems with too many collisions.

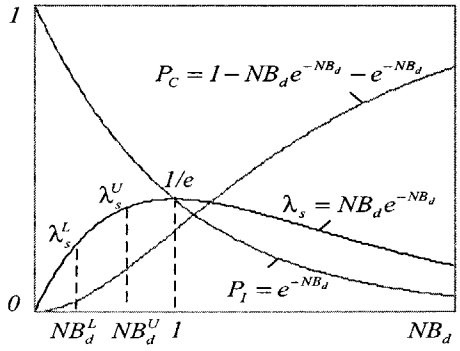


Figure 4-4: System Performance with $B_d$

Our numerical results further show that in almost all cases, these two bounds are very close to each other. Table 4.2 gives some numerical results for different system and protocol

Table 4.2: Differences between $B_d^U$ and $B_d^L$

| $D$ | $N$ | $p$ | $B_d^U$ | $B_d^L$ | $B_d^U - B_d^L$ |
|---|---|---|---|---|---|
| 20 | 20 | 0.8 | 0.0528 | 0.0509 | 0.0019 |
| 30 | 10 | 0.9 | 0.0539 | 0.0524 | 0.0015 |
| 50 | 40 | 0.7 | 0.0254 | 0.0249 | 0.0005 |

parameters $N$, $D$ and $p$. The difference between $B_d^L$ and $B_d^U$ is shown in the last column. Henceforth we approximate the actual $B_d$ by $B_d^U$ and use $B_d$ and $B_d^U$ interchangeably. In conclusion, the system performance is well approximated by Equations (4.1) and (4.10b), where Equation (4.1) expresses the system performance, and Equation (4.10b) gives the individual send rate $B_d$ needed. Note that since $f^U$ is monotonic with $Q$, the solution of Equation (4.10b) for $B_d$ is unique (see Figure 4-3).

Next consider a sufficient and necessary condition on which the maximum possible throughput $1/e$ can be achieved. Recall that this throughput can be achieved if and only if $NB_d = 1$. By plugging $NB_d = 1$ into Equation (4.10b) and noting that its solution for $B_d$ is unique, we obtain the following sufficient and necessary condition:

$$2.0N - 0.42D - RTO - \frac{0.42}{p} = 0. \tag{4.11}$$

Notice that due to the limit range of the actual parameters, the above condition cannot be always satisfied by adjusting one parameter with others fixed. For example for very large $D$, the solution of the above condition for $p$ can be negative, while the actual transmission probability $p$ has to be nonnegative. In this case, the throughput of $1/e$ cannot be achieved by adjusting $p$ only.

Although Condition (4.11) gives sufficient and necessary condition for the system throughput to achieve $1/e$, in practice, TCP updates $RTO$ based on its packet round trip time measurements, and hence we do not know the exact relationship between $RTO$ and $p$, $D$, $N$. Therefore, even when the system and protocol parameters fall into the range to satisfy Condition (4.11), from Condition (4.11) only, we cannot obtain the parameter sets that achieves throughput of $1/e$. Nevertheless, we can use the way TCP updates its $RTO$ to approximate the relationship between $RTO$ and $p$, $D$, $N$. Specifically, in TCP, $RTO$ is set to be the average round trip time seen by the transport layer packets, denoted by $RTT$, plus four times its standard deviation. Because of the large propagation delay of satellite

links, when $p$ is not close to zero, we can ignore the queuing delays the packets experience at the MAC layer and approximate $RTT$ by the sum of packet service time at the MAC layer and the round trip time. Recall that the packet service time at the MAC layer is geometrically distributed with mean $1/p$ and variance $(1-p)/p^2$. Therefore, we obtain $RTO \approx \frac{1}{p} + D + 4\frac{\sqrt{1-p}}{p}$. The parameter sets to achieve throughput of $1/e$ can then be found by plugging this expression back into Condition (4.11).

Overall, given the system and protocol parameters, the system performance can be solved from Equation (4.1) and (4.10b). Condition (4.11) is a sufficient and necessary condition for the system throughput to achieve its maximum possible value $1/e$. Due to the limited range of system and protocol parameters, this condition cannot be always satisfied by adjusting one parameter with others fixed. That is, the throughput of $1/e$ is not always achievable.

The following subsections analyze how the system performance changes with different system and protocol parameters.

### Impact of Transmission Probability on System Performance

First consider the impact of the transmission probability $p$ on the system performance. By the definition of $f^U$ (Inequality (4.8)) and noting that $RTO$ is a decreasing function of $p$, $f^U$ increases with increasing $p$. Thus as $p$ increases, curve $f^U$ in Figure 4-3 moves up. On the other hand, curve $g$ is not a function of $p$ and remains the same. Therefore, the intersection of curve $g$ and $f^U$ moves leftwards. Consequently, $B_d^U$, that is $B_d$, increases monotonically with $p$.

Recall that Condition (4.11) is a sufficient and necessary condition for $B_d$ to achieve $1/N$. Also note that $p \in [0,1]$ and when $p = 0$, $B_d = 0$. Therefore, when $p$ increases from 0 to 1, if the solution of Condition (4.11) for $p$, denoted by $p_{max}$, lies within $[0,1]$, then $B_d$ increases from 0 to a value that is larger than $1/N$. Consequently, the throughput first increases, then decreases, with maximum $1/e$ (see Figure 4-4). Otherwise, if $p_{max} \notin [0,1]$, $B_d$ increases from 0 to some value below $1/N$. Consequently, the throughput increases monotonically with $p$, and the maximum throughput is achieved at $p = 1$.

Similarly, since $P_I$ and $P_C$ is monotonic with $B_d$, $P_I$ decreases monotonically with $p$, and $P_C$ increases monotonically with $p$.

The above discussion actually gives us the transmission probability at which the through-

put achieves its highest value, denoted by $p_{opt}$, as follows:

$$p_{opt} = \begin{cases} p_{max} & \text{when } p_{max} \in [0,1] \\ 1 & \text{otherwise} \end{cases} \tag{4.12}$$

Moreover, if $p_{opt} = p_{max} \in [0,1]$, then the system throughput achieves its maximum possible value $1/e$. Otherwise, $p_{opt} = 1$, and the system performance, with throughput below $1/e$, can be solved from Equation (4.1) and Equation (4.10b).

Physically, from the perspective of the MAC layer, when $p$ is very small, the send rate $B_d$ is very small (lies in [0,1/N], see Figure 4-4). Most times the system is idle and few packets incur collisions. That is, the idle probability $P_I$ is high and the collision probability $P_C$ is low. Increasing $p$ increases $B_d$ and $P_C$ but decreases $P_I$. Although this leads to more collisions, the number of idle slots also decreases, and the overall system throughput $\lambda_s$ increases. If with increasing $p$, the send rate $B_d$ remains below $1/N$ after $p$ reaches 1, the throughput increases monotonically with $p$ (the case $p_{opt} = 1$). Otherwise, the send rate $B_d$ goes beyond $1/N$ after $p$ reaches $p_{max} < 1$, the system begins to incur too many collisions, and the throughput begins to drop. That is, in this case, the throughput first increases then decreases, with maximum $1/e$ achieved at $p_{max}$.

From the perspective of the transport layer, when $p$ is higher, on one hand, the resulting more MAC layer collisions increase the TCP packet loss probability, which should lead to smaller system throughput. On the other hand, higher $p$ reduces packet service time at the MAC layer, and the round trip time seen by the TCP packets is thus shorter. The system throughput should hence be higher. The overall system throughput is thus a tradeoff between the TCP packet loss probability and round trip time.

Figure 4-5 shows both numerical and simulation results for the throughput $\lambda_s$ as a function of $p$ when $D = 10$ and $N = 20$ as well as when $D = 50$ and $N = 20$. The numerical results are obtained by solving Equation (4.1) and (4.10b). The first case ($D = 10$) corresponds to the case $p_{opt} < 1$, and as expected, the throughput first increases with $p$ then decreases. The second case ($D = 50$) corresponds to the case $p_{opt} = 1$, and the throughput increases monotonically with $p$.

In addition, both the numerical results and simulation results for the above cases indicate that the expected collision window decreases monotonically with $p$, with value below 2 for $p \geq 0.1$. Moreover, Figure 4-5 also shows that the numerical results based on Equation (4.1)

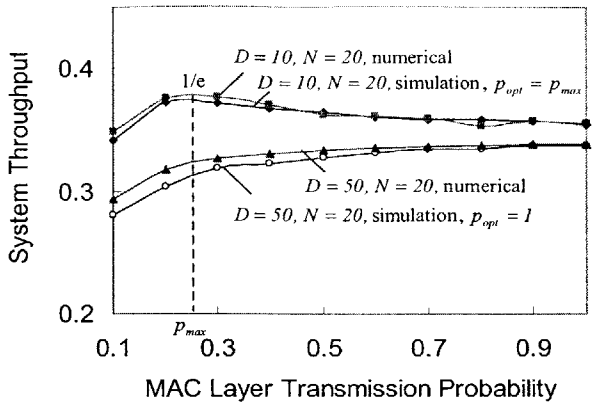


Figure 4-5: System Throughput as a Function of Transmission probability - without Backoff ($D = 10$ and $N = 20$, $D = 50$ and $N = 20$)

and (4.10b) match well with the simulation results for the TCP system, which confirms our use of the CA system to model the behavior of TCP over ALOHA as well as our analysis.

Note that since $B_d$ increases monotonically with $p$, the increase range in Figure 4-5 corresponds to systems with too many idle slots to achieve throughput of $1/e$, and the decrease range corresponds to systems with too many collisions (see Figure 4-4).

### Impact of Round Trip Time on System Performance

Analogous to the above analysis for $p$, it can be shown that $B_d$ decreases monotonically with $D$. For brevity, we omit the details. Moreover, if the range of $D$ considered includes the solution of Condition (4.11) for $D$, denoted by $D_{max}$, then the maximum possible throughput $1/e$ is achieved at $D_{max}$. In this case both larger $D$ and smaller $D$ lead to a throughout below $1/e$. Otherwise, if $D_{max}$ is outside of the range considered, the throughput is always below $1/e$.

Physically, from the MAC layer perspective, a low throughput can be either due to too many idle slots or too many collisions. For systems with $D_{max}$ falling into the range considered, the maximum system throughput is $1/e$ and is achieved at $D = D_{max}$. Starting from $D_{max}$, increasing $D$ leads to too many idle slots and reducing $D$ leads to too many collisions. Consequently, both result in monotonic decreasing of the throughput with $D$. Whereas for systems with $D_{max}$ being outside the range considered, if $D_{max}$ is above the range considered, then the system always operates in a region with too many collisions.

Increasing $D$ thus reduces collisions and hence the system throughput increases with $D$. Otherwise, if $D_{max}$ is belowe the range considered, then the system always operates in a region with too many idle slots. Increasing $D$ further increases the number of idle slots and hence the system throughput decreases with $D$.

From the perspective of the transport layer, on one hand, larger $D$ reduces collisions and packet loss probability, and the system throughput should be higher. On the other hand, larger $D$ also means longer round trip time seen by the TCP packets, and the system throughput should be lower. The overall system throughput is hence a tradeoff between TCP packet loss probability and round trip time.

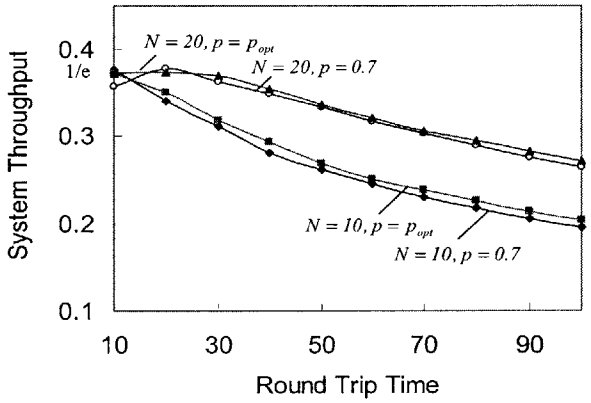


Figure 4-6: System Throughput as a Function of Round Trip Time - without Backoff ($N = 20$ and $p = 0.7$ or $p_{opt}$, $N = 10$ and $p = 0.7$ or $p_{opt}$)

Figure 4-6 shows the simulation results for the throughput $\lambda_s$ as a function of $D \in [10, 100]$ when $N = 20$ and $p = 0.7$ as well as when $N = 10$ and $p = 0.7$. In the first case ($N = 20$), the throughput $1/e$ is achieved around $D_{max} \approx 25$. Whereas in the second case ($N = 10$), $D_{max}$ is below the range considered. Even when $D = 10$, there are still too many idle slots and the throughput is still below $1/e$. Note that since $B_d$ decreases monotonically with $D$, the increase range in Figure 4-5 corresponds to systems with too many collisions to achieve throughput of $1/e$, and the decrease range corresponds to systems with too many idle slots (see Figure 4-4).

To illustrate the impact of adjusting protocol parameters on the system performance, for each $D$ in each case in Figure 4-6, $p_{opt}$ are also calculated from Condition (4.11) and Equation (4.12). Note that as mentioned before, when calculating $p_{max}$ from Condition

(4.11), we use $RTO \approx \frac{1}{p} + D + 4\frac{\sqrt{1-p}}{p}$ to approximate the TCP timeout value. The simulated throughput with $p_{opt}$ is plotted in Figure 4-6 as well. The figure shows that for large $D$, the system throughput is always below $1/e$, and $p_{opt} = 1$. Whereas when $D$ is relatively small, the highest possible throughput $1/e$ can always be achieved by setting the transmission probability $p = p_{opt} = p_{max}$. That is, we can always lower the transmission probability $p$ to counterbalance the collisions resulting from small $D$. Notice that using $p_{opt}$ obtained by using $RTO \approx \frac{1}{p} + D + 4\frac{\sqrt{1-p}}{p}$ can indeed give us throughput close to $1/e$.

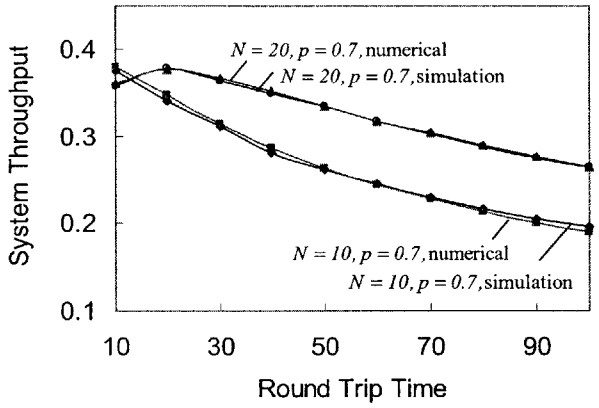


Figure 4-7: Comparison between Numerical and Simulation Results - Throughput vs Round Trip Time - without Backoff
($N = 20$ and $p = 0.7$, $N = 10$ and $p = 0.7$)

Numerical results with the same sets of parameters are calculated from Equation (4.1) and (4.10b) and plotted in Figure 4-7. For comparison purpose, the simulation results plotted in Figure 4-6 are replotted here as well. Again, the figure shows good match between the numerical results and the simulation results.

Moreover, the numerical and simulation results for all cases also show that the expected collision window is a nondecreasing function of $D$ and is always below 3. Actually even when $D$ is further increased to 5000, the expected collision window is still below 8, which justifies our use of the CA system to model the behavior of TCP over ALOHA.

### Impact of Number of Users on System Performance

The analysis for the impact of the number of users $N$ on the system performance is also analogous to that for the transmission probability $p$. Specifically, when $N$ increases, curve

$f^U$ remains the same while curve $g$ moves downwards. Consequently, the intersection of the two curves moves leftwards, and $Q^L$ decreases. Recall that $NB_d$ increases as $Q$ decreases (Equation (4.2)). Therefore, $NB_d$ increases monotonically with $N$.

In addition, it can be easily verified that, for any $p \in [0, 1]$, $D \geq 1$ and $RTO \geq \frac{1}{p} + D$ (the actual $RTO$ should be at least the average round trip time of TCP packets, which is at least $\frac{1}{p} + D$), the solution of Condition (4.11) for $N$, denoted by $N_{max}$, is always greater than 1. Therefore, as $N$ ranges from 1 to infinity, $NB_d$ increases from some value below 1 to infinity. Consequently, the system throughput first increases then decreases, with maximum close to $1/e$ achieved at the integer closest to $N_{max}$.

Physically, from the perspective of the MAC layer, larger $N$ makes the packets in the system more "dense", which means less idle slots and more collisions. When $N$ is close to $N_{max}$, the idle slots and collisions reach a balance and the throughput is close to its highest value $1/e$. Further decreasing $N$ or increasing $N$ results in either too many idle slots, or too many collisions, both of which lower the throughput.

From the perspective of the transport layer, although the more collisions due to larger $N$ increase TCP packet loss probability and reduce the individual throughput of each SD pair, the overall system throughput is $N$ times the individual throughput. As a result, the system throughput still increases with $N$ when $N$ is small. However, when $N$ is already large, the system throughput drops with $N$.

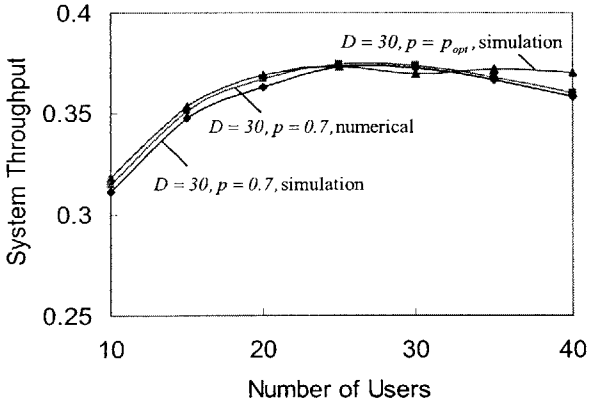


Figure 4-8: System Throughput as a Function of the Number of SD Pairs - without Backoff ($D = 30$ and $p = 0.7$, $D = 30$ and $p = p_{opt}$)

Figure 4-8 shows the simulation results for the throughput $\lambda_s$ as a function of $N \in$

$[10, 40]$ when $D = 30$ and $p = 0.7$ as well as when $D = 30$ and $p = p_{opt}$. Again, as mentioned before, $p_{opt}$ is calculated from Condition (4.11) and Equation (4.12) by setting $RTO \approx \frac{1}{p} + D + 4\frac{\sqrt{1-p}}{p}$. The throughput of the curve with $p = 0.7$ has maximum value of $1/e$, and first increases then decreases, which confirms the above analysis. The curve with $p = p_{opt}$ further shows that for large enough $N$, the maximum possible throughput $1/e$ can be achieved by adjusting the transmission probability according to Condition (4.11). That is, in order to achieve higher throughput, we can always lower the transmission probability $p$ to counterbalance the collisions resulting from large $N$. Note that since $NB_d$ increases monotonically with $N$, the increase range in Figure 4-8 corresponds to systems with too many idle slots to achieve throughput of $1/e$, and the decrease range corresponds to systems with too many collisions.

Numerical results with $D = 30$ and $p = 0.7$ are also computed and plotted in Figure 4-8. Again, they match well with the simulation results. In addition, both numerical results and simulations give the expected collision window size in all cases being below 2, which justifies our conjecture of small TCP window size and the approximation of the TCP system by the CA system.

In summary, the numerical curves obtained from the equations match well with the simulation results for the TCP system, and the equations can be used to describe the performance of the TCP system. In particular, given the system and protocol parameters, $p$, $D$ and $N$, Equation (4.1) and (4.10b) give us the system performance. Condition (4.11) is a sufficient and necessary condition on the parameters for the system to achieve its maximum possible throughput $1/e$. The optimal transmission probability at which the throughput can achieve its highest value is given in Equation (4.12). For systems with very small $D$ and/or very large $N$, the throughput $1/e$ can always be achieved by setting $p$ to its optimal value. For fixed $p$, a system with very large $D$ and/or very small $N$ has a smaller throughput than a system with relatively smaller $D$ and/or larger $N$ due to too many idle slots. On the contrary, a system with very large $N$ has a smaller throughput than a system with relatively smaller $N$ due to too many collisions. Furthermore, due to the MAC layer collisions caused by random access, in all cases in our numerical computations and simulations, the expected collision window is below 8, which justifies our use of the CA system to model the behavior of TCP over ALOHA.

## 4.4 Systems with TCP Timeout Backoff

In real TCP protocols, TCP *RTO* has exponential backoff and Karn's algorithm is applied as well. Specifically, TCP timeout backoff specifies that, after a timeout and before an acknowledgement is received for a packet that was not retransmitted, *RTO* is doubled each time a timeout occurs until it is 64 times of its original value or it reaches its upper limit. In addition, Karn's algorithm says that TCP round trip time estimates and *RTO* should not be updated upon an ACK for a packet that was retransmitted. Figure 4-9 illustrates timeout backoff and Karn's algorithm, where the white boxes indicate successful transmissions and the gray boxes indicate unsuccessful transmissions. Note that due to Karn's algorithm, although there are successful transmissions (the third transmission of packet 2 in Figure 4-9), *RTO* may still be backed off.

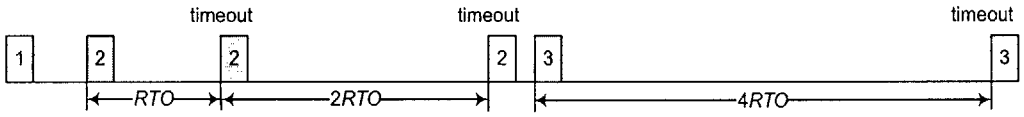


Figure 4-9: Illustration of TCP Timout Backoff and Karn's Algorithm

The analysis approach for systems with TCP timeout backoff is similar to that for systems without TCP Timeout Backoff and Karn's Algorithm. Specifically, the system level analysis is exactly the same. Consequently, Equation (4.1) and (4.2) still hold for systems with timeout backoff and Karn's algorithm, and the maximum possible throughput is $1/e$ as well. The difference between the analysis of the two types of systems lies in the session level analysis. As will be shown later, this difference leads to significantly different system performance.

This section first presents the session level analysis for systems with timeout backoff and Karn's algorithm, then discusses the system performance.

### 4.4.1 Session Level Analysis

Similar to the system without *RTO* backoff in previous section, the effects of the other $N-1$ SD pairs on one particular SD pair are aggregated into one parameter $Q$, the idle probability of all other $N-1$ SD pairs. This particular SD pair can thus be modelled as a transport layer session with loss probability of each packet being $1-Q$. Again, since in our

system all TCP packets must pass the MAC layer queuing system, the TCP session here is much more complicated than the TCP sessions analyzed in literatures [43, 36]. We hence analyze the CA system instead of the TCP system, and use the property of the CA system to model one particular SD pair as a renewal process.

Specifically, for a timeout interval between two successive timeouts, define the effective *RTO* of the interval to be the *RTO* that results in the termination of this interval. In addition, for a timeout interval, if its effective *RTO* is not backed off from the effective *RTO* of its previous timeout interval, then the timeout interval is called non-backoff interval. Otherwise, it is called backoff interval. Recall that due to our large TCP timeout assumption, upon the start of each non-backoff interval, the system state is the same. That is, there are no packets at the MAC layer. Moreover, the window size and *RTO* has the exactly the same evolution afterwards. Therefore, statistically, the session has the same behavior after the start of each non-backoff interval. The session can thus be modelled as a renewal process, with renewal points being the start of each non-backoff interval. One renewal cycle hence consists of one non-backoff interval and a sequence of backoff intervals originated from the non-backoff interval. Note that if the non-backoff interval is not backed off, then the cycle has only one interval, that is the non-backoff interval. Figure 4-10 illustrates one renewal cycle, with (a) corresponding to the case without backoff intervals, and (b) corresponding to the case with two backoff intervals.

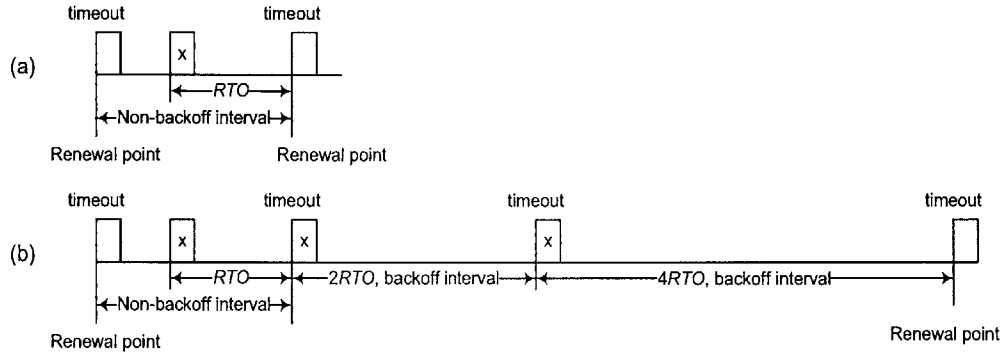


Figure 4-10: One Renewal Cycle for TCP Session with *RTO* Backoff
(a) no backoff case; (b) two backoff case

Note that different from systems without TCP timeout backoff, the renewal point of current system is the start of each non-backoff interval instead of each timeout interval. The

reason is that here due to $RTO$ backoff, effective $RTO$s of backoff intervals have different statistics from those of non-backoff intervals. The session hence behaves differently within these two types of intervals.

Within one renewal cycle, let $N_b$ denote the number of timeout intervals, $M_k$ denote the number of packets sent in the $k$th interval, and $T_k$ denote the length of the $k$th interval, with $k = 1, .., N_b$. Then by renewal theory,

$$B_d = \frac{E[\sum_{k=1}^{N_b} M_k]}{E[\sum_{k=1}^{N_b} T_k]}. \tag{4.13}$$

In the following, we first consider the condition of TCP backoff and the distribution of $N_b$, then derive $E[\sum_{k=1}^{N_b} M_k]$ and $E[\sum_{k=1}^{N_b} T_k]$.

Consider an arbitrary timeout interval. Let $L$ denote the number of packets that need to be retransmitted, and $Z$ denote the index of the first packet that incurs a collision. Recall that we assume TCP retransmits only packets that incur collisions. Hence from the rules of TCP timeout backoff and the Karn's algorithm, the effective $RTO$ of this interval is backed off from the effective $RTO$ of the previous interval if and only if at least one of the retransmissions of the $L$ packets or the transmission of the first new packet incurs collision, that is, $L + 1 \geq Z$. Recall that each packet incurs a collision with probability $1 - Q$, and the distribution of $Z$ is $Pr(Z = z) = Q^{z-1}(1 - Q)$. Therefore, the probability that this interval is a backoff interval, called backoff probability and denoted by $P_b$, is as follows:

$$P_b = \sum_{Z=1}^{L+1} Q^{Z-1}(1 - Q) = 1 - Q^{L+1}. \tag{4.14}$$

The above equation shows that the backoff probability is a function of $L$, which is further a function of packet transmission patterns of previous timeout intervals. Clearly, due to different collision patterns of timeout intervals, $L$ is not necessarily the same for all intervals. In addition, since each interval starts with a timeout, we have $L \geq 1$. Note that in [44], the backoff probability is approximated as $1 - Q$, which is different from that in Equation (4.14). The reason is that the authors there assumed that $L = 1$ and did not take into account the fact that the loss of the first new packet will also result in $RTO$ backoff.

For simplicity, we approximate $L$ for all intervals to be the same. The backoff probability is hence the same for all intervals. From the definition of renewal cycle, the number of timeout intervals within one renewal cycle $N_b$ is thus geometrically distributed with the

following distribution and expectation:

$$\begin{aligned} Pr(N_b = n) &= P_b^{n-1}(1 - P_b) \text{ with } n \geq 1, \\ E[N_b] &= \frac{1}{1 - P_b}. \end{aligned} \tag{4.15}$$

Next consider $E[\sum_{k=1}^{N_b} M_k]$. Let $M$, $M_{nb}$ and $M_b$ denote the number of packets sent during an arbitrary timeout interval, a non-backoff interval and a backoff interval, respectively. Then $E[M] = E[M_{nb}](1 - P_b) + E[M_b]P_b$. Recall that in a renewal cycle, the first interval is a non-backoff interval and the rest are backoff intervals. Hence $E[M_1] = E[M_{nb}]$. Also note that we approximate $L$ to be the same for all intervals. The packet transmission patterns are thus statistically the same for all backoff intervals. Hence, for $k = 2, .., N_b$, the expectations of all $M_k$'s are the same and equal to $E[M_b]$. We therefore have

$$\begin{aligned} E[\sum_{k=1}^{N_b} M_k] &= E[M_1] + E[\sum_{k=2}^{N_b} M_k] \\ &= E[M_{nb}] + E[N_b - 1]E[M_b] \\ &= E[M_{nb}] + (\frac{1}{1 - P_b} - 1)E[M_b] \\ &= \frac{(1 - P_b)E[M_{nb}] + P_b E[M_b]}{1 - P_b} \\ &= \frac{E[M]}{1 - P_b}. \end{aligned} \tag{4.16}$$

Moreover, for an arbitrary interval, for both sessions with and without TCP timeout backoff, the window evolution and packet transmission patterns are statistically the same. Hence $E[M]$ in Equation (4.6) derived for sessions without TCP timeout backoff is also applicable to sessions with TCP timeout backoff. By plugging it into the above equation, we obtain

$$E[\sum_{k=1}^{N_b} M_k] = \frac{\frac{1}{1-Q} + \sum_{k=1}^{\infty} Q^{k(k+1)/2}}{1 - P_b}. \tag{4.17}$$

To obtain $E[\sum_{k=1}^{N_b} T_k]$, we need one more quantity: $E[\sum_{k=1}^{N_b} RTO_k]$, where $RTO_k$ is defined to be the effective $RTO$ of the $k$th interval within one renewal cycle. Further denote

$RTO$ to be the TCP average timeout value excluding all backoffs. Then $E[RTO_1] = RTO$. From the rules of TCP exponential backoff and Karn's algorithm, for different $k$, $RTO_k$ is as follows:

$$RTO_k = \begin{cases} 2^{k-1}RTO & \text{for } k \leq 7 \\ 64RTO & \text{for } k > 7 \end{cases} \tag{4.18}$$

Note that here for simplicity, we allow $RTO_k$ to go beyond its maximum value. After combining the above equation with Equation (4.15) and some manipulations, we obtain $E[\sum_{k=1}^{N_b} RTO_k]$ as follows:

$$\begin{aligned} E[\sum_{k=1}^{N_b} RTO_k] &= \sum_{n=1}^{\infty}(\sum_{k=1}^{n} RTO_k)P_b^{n-1}(1-P_b) \\ &= RTO(\sum_{n=1}^{7}(\sum_{k=1}^{n} 2^{k-1})P_b^{n-1}(1-P_b) + (\sum_{k=1}^{7} 2^{k-1} + \sum_{k=8}^{n} 64)P_b^{n-1}(1-P_b)) \\ &= RTO\frac{1+P_b+2P_b^2+4P_b^3+8P_b^4+16P_b^5+32P_b^6}{1-P_b} \\ &= RTOf(P_b)/(1-P_b) \\ &= RTOf(1-Q^{L+1})/(1-P_b). \end{aligned} \tag{4.19}$$

where $f(x)$ is a function defined to be $f(x) = 1 + x + 2x^2 + 4x^3 + 8x^4 + 16x^5 + 32x^6$, and the last equality comes from Equation (4.14).

$E[\sum_{k=1}^{N_b} T_k]$ can then be derived in a similar way as that for $E[\sum_{k=1}^{N_b} M_k]$. Specifically, let $T$, $T_{nb}$ and $T_b$ denote the duration of an arbitrary timeout interval, a non-backoff interval and a backoff interval, respectively. Let $RTO$, $RTO_{nb}$ and $RTO_b$ denote their effective $RTO$, and $t$, $t_{nb}$ and $t_b$ denote the durations of these interval excluding the corresponding $RTO$s. Then $T = t + RTO$, $T_{nb} = t_{nb} + RTO_{nb}$, $T_b = t_b + RTO_b$, and $E[t] = E[t_{nb}](1-P_b) + E[t_b]P_b$. Since the packet transmission patterns are statistically the same for all backoff intervals except that they may have different effective RTOs, we have $E[t_1] = E[t_{nb}]$ and for $k = 2, .., N_b$, the expectation of $t_k$ equals to $E[t_b]$. Hence $E[\sum_{k=1}^{N_b} T_k] = E[\sum_{k=1}^{N_b} t_k] + E[\sum_{k=1}^{N_b} RTO_k] = \frac{E[t]}{1-P_b} + RTOf(1-Q^{L+1})/(1-P_b) = \frac{E[t]+RTOf(1-Q^{L+1})}{1-P_b}$, where the second equality is derived by exactly the same approach as the derivation in Equation (4.17). Therefore, from the bounds of $E[T]$ in Equation (4.7), which is derived for sessions without TCP timeout backoff

but is also applicable for $E[t]$ here, we can directly obtain the following bounds for the session considered:

$$\begin{aligned}&\frac{(\frac{1}{p}+D)\sum_{k=1}^{\infty}Q^{k(k+1)/2}+RTOf(1-Q^{L+1})}{1-P_b}\leq E[\sum_{k=1}^{N_b}T_k]\\&\leq\frac{\frac{1}{p(1-Q)}+(\frac{1}{p}+D)\sum_{k=1}^{\infty}Q^{k(k+1)/2}+RTOf(1-Q^{L+1})}{1-P_b}.\end{aligned}\tag{4.20}$$

We now have $E[\sum_{k=1}^{N_b}M_k]$ in Equation (4.17) and bounds for $E[\sum_{k=1}^{N_b}T_k]$ in Inequality (4.20). By plugging them into Equation (4.13), we obtain the following bounds for $B_d$ in terms of $Q$:

$$\begin{aligned}&\frac{\frac{1}{1-Q}+\sum_{k=1}^{\infty}Q^{k(k+1)/2}}{\frac{1}{p(1-Q)}+(\frac{1}{p}+D)\sum_{k=1}^{\infty}Q^{k(k+1)/2}+RTOf(1-Q^{L+1})}\leq B_d\\&\leq\frac{\frac{1}{1-Q}+\sum_{k=1}^{\infty}Q^{k(k+1)/2}}{(\frac{1}{p}+D)\sum_{k=1}^{\infty}Q^{k(k+1)/2}+RTOf(1-Q^{L+1})}.\end{aligned}\tag{4.21}$$

Note that the only difference between the above two bounds is that the denominator of the lower bound has one extra term $\frac{1}{p(1-Q)}$.

### 4.4.2 System Performance

Based on the system level analysis and session level analysis, this subsection first gives formula for computing the system performance as well as a sufficient and necessary condition for the system throughput to achieve its maximum value 1/e, then discusses the impacts of different system and protocol parameters on the system performance.

#### System Performance and Condition for Maximum Throughput

For the current system considered, the analysis of system performance, is analogous to that for systems without TCP timeout backoff, as given in Section 4.3.3. Specifically, the system level analysis gives one relationship between $B_d$ and $Q$, that is $Q \approx e^{-NB_d}$. The session level analysis gives lower and upper bounds for $B_d$ in terms of $Q$, as in Inequality (4.21). Combining them together gives lower and upper bounds for $B_d$ in terms of system and protocol parameters, which further gives the range of system performance, that is, system

throughput, idle probability and collision probability (see Section 4.3.3). In particular, by plugging $Q = e^{-NB_d}$ into the right hand side of Inequality (4.21), the upper bound for $B_d$ in terms of system and protocol parameters is the solution of the following equation:

$$B_d[(\frac{1}{p} + D)\sum_{k=1}^{\infty} e^{-NB_d k(k+1)/2} + RTOf(1 - e^{-(L+1)NB_d})] = \frac{1}{1 - e^{-NB_d}} + \sum_{k=1}^{\infty} e^{-NB_d k(k+1)/2}. \tag{4.22}$$

For different combinations of system and protocol parameters, our numerical computations show that the lower and upper bounds of $B_d$ obtained from $Q \approx e^{-NB_d}$ and Inequality (4.21) are very close to each other. We hence use the upper bound, which is the solution of Equation (4.22), to approximate $B_d$. The system performance can then be directly computed by Equation (4.1) and (4.22). In particular, the system throughput is $\lambda_s \approx NB_d e^{-NB_d}$.

A sufficient and necessary condition for the system throughput to achieve its maximum value 1/e can also be derived from Equation (4.22). Specifically, the system throughput can achieve $1/e$ if and only if the attempt rate $NB_d = 1$ (see Equation (4.1)). After plugging $NB_d = 1$ into Equation (4.22) and noting that the solution of Equation (4.22) for $B_d$ is unique, the sufficient and necessary condition is hence as follows:

$$2.0N - 0.42D - 31.52RTO - \frac{0.42}{p} = 0. \tag{4.23}$$

Here when computing $f(1 - e^{-(L+1)NB_d})$, we use $L = 1$. The reason is that, due to the long duration of the TCP backoff periods, we expect that the system send rate is not high and the packet loss probability is not high either. Hence the number of packets needed to be retransmitted within one interval is low, and one should thus be a reasonable value of $L$. This is evidenced by the simulation results presented later.

Unfortunately, due to the large coefficient of $RTO$, this condition can be satisfied only for very large $N$. Specifically, according to TCP $RTO$ computation, $RTO > 1/p + D$. The transmission probability at the MAC layer is $p \in [0, 1]$. Therefore, to satisfy Condition (4.22), the number of SD pairs $N$ needs to satisfy $N > 15.97(1/p+D)$. Clearly this cannot be satisfied by most systems, especially systems with large bandwidth-delay product (large $D$, which is typical in satellite systems, and small transmission probability $p$). The maximum

throughput 1/e is hence not reachable for most systems, as evidenced by our numerical and simulation results given later.

In summary, given the system and protocol parameters, the system performance can be solved from Equation (4.1) and (4.22). Condition (4.23) is a sufficient and necessary condition for the system throughput to achieve its maximum possible value $1/e$. Unfortunately, this condition cannot be satisfied by most systems except for those with very large number of users.

For general systems where $N$ is not very large, the following subsections discusses how the system performance changes with different system and protocol parameters.

### Impact of System and Protocol Parameters on System Performance

Similar to the curves in Figure 4-3, we can plot curve $g$ and bounds of $B_d$ in terms of $Q$ as in Inequality (4.21). Then by similar analysis as that in Section 4.3.3, we can see that $NB_d$ increases with $p$ and $N$ and decreases with $D$. To avoid duplications, we omit the details here.

In addition, as discussed previously, $NB_d = 1$ cannot be achieved by general systems except for those with very large $N$. Recall that $NB_d$ increases with $N$. Hence general systems operate at a range with $NB_d \ll 1$, that is, the monotonic increasing region of throughput in Figure 4-4. Therefore, the fact that $NB_d$ increases with $p$ and $N$ means that the system throughput also increases with $p$ and $N$, while decreasing $NB_d$ with $D$ means that the system throughput decreases with $D$. The optimal transmission probability where maximum system throughput is achieved is thus one.

Physically, MAC layer collisions lead to high TCP packet loss probability, which further results in TCP timeout backoffs. Few packets are transmitted during backoff intervals. The system hence has too many idle slots, and the throughput is thus low. Accordingly, increasing MAC layer transmission probability and the number of users reduces the number of idle slots. Since the system operation range is limited to be within the monotonic increasing region, this reduction of the number of idle slots will not be accompanied by too many collisions. Hence the system throughput increases with $p$ and $N$. On the contrary, increasing the round trip time deteriorates the system throughput by further increases the number of idle slots. Consequently, the system throughput decreases with $D$.

From the perspective of the transport layer, on one hand, higher $p$ and lower $D$ means

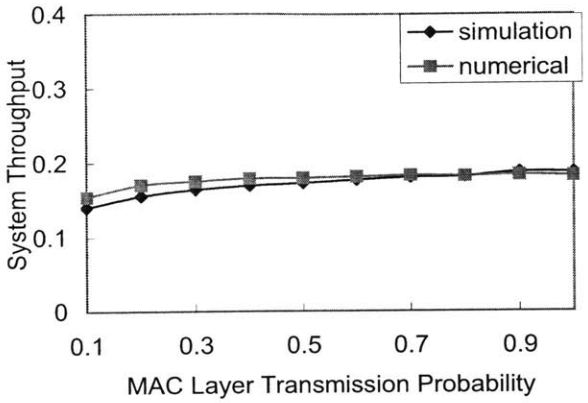


Figure 4-11: System Throughput as a Function of Transmission Probability - with Backoff ($N = 10$ and $D = 50$)

more MAC layer collisions and larger TCP packet loss probability, which should lead to a lower system throughput. On the other hand, higher $p$ and lower $D$ also means smaller round trip time seen by the TCP packets, which should lead to a higher system throughput. The overall system throughput is thus a tradeoff between the TCP packet loss probability and round trip time.

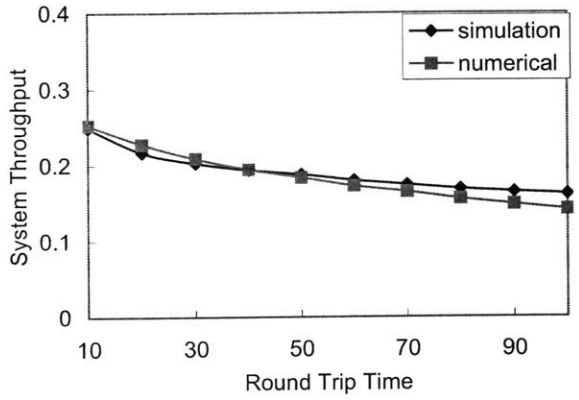


Figure 4-12: System Throughput as a Function of Round Trip Time - with Backoff ($N = 10$ and $p = 0.9$)

Figure 4-11, 4-12 and 4-13 plot both numerical and simulated system throughput as a function of $p$, $D$ and $N$, respectively, where Figure 4-11 has $N = 10$ and $D = 50$, Figure 4-12 has $p = 0.9$ and $N = 10$, and Figure 4-13 has $p = 0.9$ and $D = 100$. The numerical

throughput is computed from Equation (4.22) and (4.1). The curves confirm the above discussions, that is, the system throughput increases with $p$ and $N$ and decreases with $D$. In addition, the figures show that the system throughput is much lower than $1/e$. This is consistent with previous discussions, that is, Condition (4.23) is not satisfied by the systems examined.

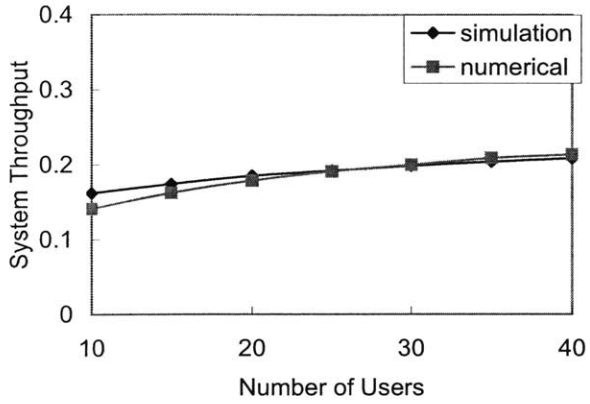


Figure 4-13: System Throughput as a Function of the Number of SD Pairs - with Backoff ($D = 100$ and $p = 0.9$)

Furthermore, the figures also indicate good match between numerical and simulation results, which confirms the accuracy of Equation (4.1) and (4.22). All simulations and numerical results also yield collision window size below 6. To further validate the approximation of the TCP system by the CA system, Figure 4-14 plots the system throughput as a function of transmission probability with $N = 10$ and $D = 50$. It can be seen that the system throughput of the two systems are indeed very close to each other.

For completeness, the optimal transmission probability is derived from Condition (4.23). Specifically,

$$p_{opt} = \begin{cases} p_{max} & \text{when } p_{max} \in [0,1] \\ 1 & \text{otherwise} \end{cases} \tag{4.24}$$

where $p_{max}$ is the solution of Condition (4.23). Moreover, if $p_{opt} = p_{max} \in [0,1]$, then the system throughput achieves 1/e. Otherwise, $p_{opt} = 1$, and the system performance, with throughput below $1/e$, can be solved from Equation (4.1) and Equation (4.22). Note that as discussed previously, for most systems, the solution of Condition (4.23) $p_{max} \notin [0,1]$, and

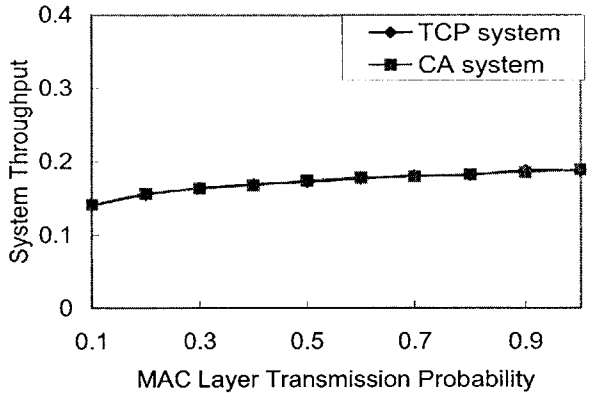


Figure 4-14: Comparison between CA System and TCP system - with Backoff ($N = 10$ and $D = 50$)

throughput of 1/e is not achievable. $p_{opt}$ is thus one for these systems.

## 4.5 Impacts of TCP Timeout Backoff and MAC Layer Retransmissions

For systems without MAC layer retransmissions, the previous two sections analyze systems with and without TCP timeout backoff, respectively. We can see that their performance is significantly different. This section discusses in detail the impacts of TCP timeout backoff on system performance, as well as the impacts of MAC layer retransmissions.

### 4.5.1 Impacts of TCP Timeout Backoff

For systems with and without TCP timeout backoff, the previous two sections give two equations for solving the system performance, Equation (4.1), (4.10b) and Equation (4.1), (4.22), respectively. For clarity, we rewrite them here:
for systems without TCP timeout backoff:

$$B_d[(\frac{1}{p} + D)\sum_{k=1}^{\infty} e^{-NB_d k(k+1)/2} + RTO] = \frac{1}{1 - e^{-NB_d}} + \sum_{k=1}^{\infty} e^{-NB_d k(k+1)/2}, \quad (4.25a)$$

$$\lambda_s \approx NB_d e^{-NB_d}, \quad (4.25b)$$

for systems with TCP timeout backoff:

$$B_d[(\frac{1}{p}+D)\sum_{k=1}^{\infty}e^{-NB_dk(k+1)/2}+RTOf(1-e^{-(L+1)NB_d})]=\frac{1}{1-e^{-NB_d}}+\sum_{k=1}^{\infty}e^{-NB_dk(k+1)/2}, \tag{4.26a}$$

$$\lambda_s \approx NB_de^{-NB_d}, \tag{4.26b}$$

For convenience, we denote the solutions of Equation (4.25a) and (4.26a) by $B_d^{nb}$ and $B_d^b$, respectively.

By comparing Equation (4.25a) and (4.26a), we see that their only difference lies in the different coefficient of $RTO$, where for systems without backoff the coefficient is one and for systems with backoff it is $f(1-e^{-(L+1)NB_d})$. Recall that the function $f(x)$ is defined to be $f(x)=1+x+2x^2+4x^3+8x^4+16x^5+32x^6$ and $f(x)>1$ for all positive $x$. Moreover, $B_d^{nb}$ and $B_d^b$ are intersections of corresponding curves in Figure 4-3. Hence the curve $f^U$ corresponding to systems with backoff is lower than that corresponding to systems without backoff. Consequently, $B_d^b < B_d^{nb}$. That is, the system with TCP timeout backoff has a lower individual send rate than the system without TCP timeout backoff.

Furthermore, the discussions in Section 4.4.2 shows that general systems without very large $N$ that employs timeout backoff operates within a range with $NB_d \ll 1$. Hence from Equation (4.25) and (4.26), systems with backoff has a lower system throughput than systems without backoff. For these two types of systems with $N=10$ and $D=50$, Figure 4-15 plots system throughput as a function of MAC layer transmission probability. As expected, the figure clearly shows a lower throughput of systems with backoff.

Physically, for systems with TCP backoff, the long duration of TCP backoff periods leads to significantly low send rate and too many idle slots. The system hence operates in a range with $NB_d \ll 1$, and the throughput is far below $1/e$ (see Figure 4-4). Whereas for systems without backoff, the send rate is higher, which results in higher throughput.

Another difference between systems with and without backoff is the condition for systems to achieve the maximum possible throughput $1/e$, Condition (4.11) and (4.23), respectively. For clarity, we rewrite them here:

for systems without TCP timeout backoff:

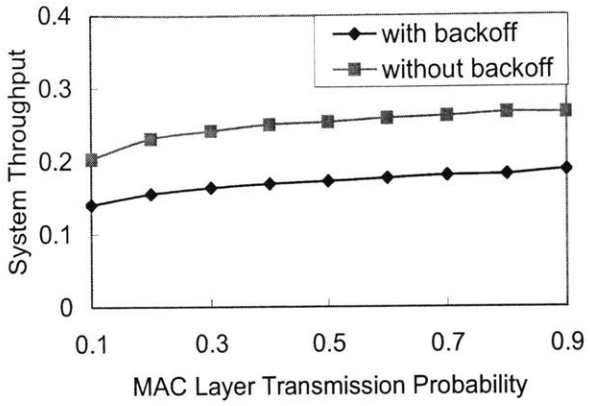


Figure 4-15: System Throughput as a Function of Transmission Probability - Comparison of Systems with and without Backoff
($N = 10$ and $D = 50$)

$$2.0N - 0.42D - RTO - \frac{0.42}{p} = 0. \tag{4.27}$$

for systems with TCP timeout backoff:

$$2.0N - 0.42D - 31.52RTO - \frac{0.42}{p} = 0. \tag{4.28}$$

We can see that the coefficient of $RTO$ for systems with backoff is much higher than that for systems without backoff. This large coefficient leads to the result that the condition can be satisfied only for systems with very large $N$. Consequently, system throughput cannot be achieved for general systems without very large number of users. Whereas for systems without backoff, the condition can be satisfied by more systems. Consequently, in some cases (small $D$ and large $N$), the throughput of $1/e$ can be achieved by adjusting the MAC layer transmission probability $p$.

Moreover, the two different conditions also lead to the fact that for general systems, the throughput of systems with backoff is monotonic with system and protocol parameters $p$, $D$ and $N$, as evidenced by Figure 4-11, 4-12 and 4-13, while the throughput of systems without backoff may be concave, as shown in Figure 4-5, 4-6 and 4-8. The summits of the concave curves correspond to the parameter sets that satisfy Condition (4.27).

Physically, for systems with TCP backoff, due to the long duration of the backoff periods, even when the MAC layer transmission probability $p$ is one, there are still too many idle

slots. Recall that the send rate increases monotonically with $p$. Hence the attempt rate $NB_d$ cannot reach one for all $p \in [0, 1]$, and Condition (4.28) cannot be satisfied.

The above discussions show that TCP timeout backoff leads to significant different system performance. Mathematically, the difference is due to the large coefficient of $RTO$ in Equation (4.26a) and Condition (4.28). Physically, the difference is due to the long duration of the TCP backoff periods. The overall system throughput is hence much lower than that of systems without backoff.

### 4.5.2 Impacts of MAC Layer Retransmissions

The ALOHA scheme with retransmissions follows both of the two rules described in Section 4.2 plus the following additional rule:

- Upon a collision, the sources involved will receive feedback from the satellite after an average of $D/2$ slots. The collided packet then rejoins the queue after all packets waiting for retransmissions and before any new packets. Each packet can be transmitted up to $r$ times. If none of the $r$ transmissions are successful, the packet will be discarded.

Here we assume that the distance between each source and the satellite is approximately the same as that between the satellite and each destination, which is reasonable for satellite links. Hence the round trip time between each source and the satellite is a random variable with mean $D/2$ slots.

According to the above scheme, each source MAC layer can be viewed as a queuing system with feedback. The queue has all arrivals coming from its corresponding transport layer and geometrically distributed service time with mean $1/p$. Also note that if $r = 1$, the scheme does not allow MAC layer retransmissions of collided packets. The queue then becomes a standard queue without feedback. Moreover, in this case, TCP retransmission is the only way to recover collided packets, and the system is reduced to the system analyzed previously.

Similar to systems without MAC layer retransmissions discussed previously, for systems with MAC layer retransmissions, we have definitions of send rate and throughput at both the transport layer and the MAC layer. In addition, since MAC collision losses are the only losses in the system, the throughput at both layers are the same. However, due

to the MAC layer retransmissions, the send rates at both layers are no longer the same. Nevertheless, if we denote the individual send rate at the MAC layer by $B_d$, then from the independence assumption between users, Equation (4.1) still holds. Hence, the number of packets correctly received in a slot can still be approximated by a Poisson random variable, and the maximum possible throughput is $1/e$ as well. Moreover, either too many idle slots or too many collisions result in a lower throughput.

However, due to the complicated MAC layer queue system with feedback, the session level analysis is much more complicated. We hence use simulations instead of analysis to examine the system performance.

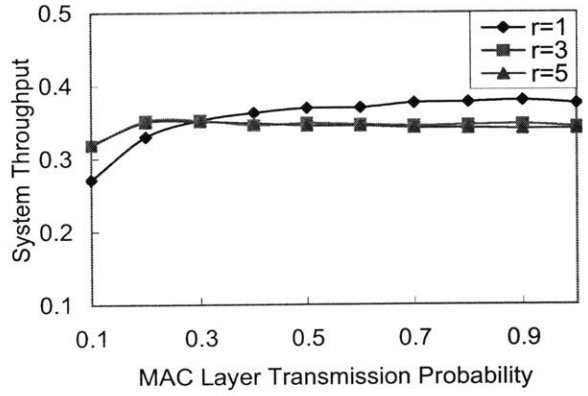


Figure 4-16: System Throughput vs Transmission Probability for Different $r$ - without Backoff

($N = 10$ and $D = 10$)

First consider systems without TCP backoff. For values of $r = 1$, 3 and 5, Figure 4-16, 4-17 and 4-18 plot simulated throughput as a function of transmission probability $p$, round trip time $D$, and number of users $N$, respectively. The parameters used in the three figures are $D = 10$ and $N = 10$, $p = 0.7$ and $N = 10$, $p = 0.7$ and $D = 30$, respectively. The figures show that MAC layer retransmissions (corresponding to curves with $r = 3$ and $r = 5$) improve the system throughput when $p$ is small, $N$ is small, and/or $D$ is large. Recall that these scenarios corresponds to systems with low $NB_d$ and too many idle slots, that is, systems with light load. Hence MAC retransmissions are preferred for systems with light load.

In practice, TCP needs around $RTO$ slots to detect collisions and react accordingly, while the MAC layer needs around $D/2$ slots, which is much shorter than $RTO$. Hence

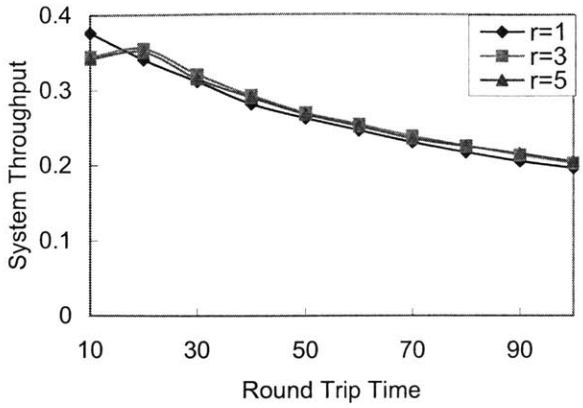


Figure 4-17: System Throughput vs Round Trip Time for Different $r$ - without Backoff ($N = 10$ and $p = 0.7$)

the MAC layer reacts to collisions much faster than TCP. Consequently, compared to TCP retransmissions, MAC layer retransmissions inject more traffic into the shared channel, and hence lead to higher send rate. Moreover, this higher send rate results in more collisions, which further trigger MAC layer retransmissions and leads to even higher send rate. As a result, systems with MAC layer retransmissions have much higher send rate than systems without MAC layer retransmissions. Therefore, from Equation (4.1), MAC layer retransmissions are preferred for systems with light load. In contrast, when the system is heavily loaded, retransmissions at the transport layer (TCP) result in higher throughput.

From the perspective of the MAC layer, the above behavior is because the MAC layer retransmissions increase collisions and reduce idle slots, hence performs well for systems with light load but performs poorly for systems with heavy load. From the perspective of the transport layer, on one hand, MAC layer retransmissions recover some collided packets and thus reduce TCP packet loss probability, which should lead to an increase the system throughput. On the other hand, MAC layer retransmissions also increase the round trip time seen by the TCP packets, which should lead to a decrease of the system throughput. As a result, the system performance is a tradeoff between the TCP packet loss probability and round trip time.

Next consider systems with TCP backoff. With the same sets of parameters as the corresponding figure among Figure 4-16 to 4-18, for values of $r = 1$, 3 and 5, Figure 4-19, 4-20 and 4-21 plot simulated throughput as a function of transmission probability

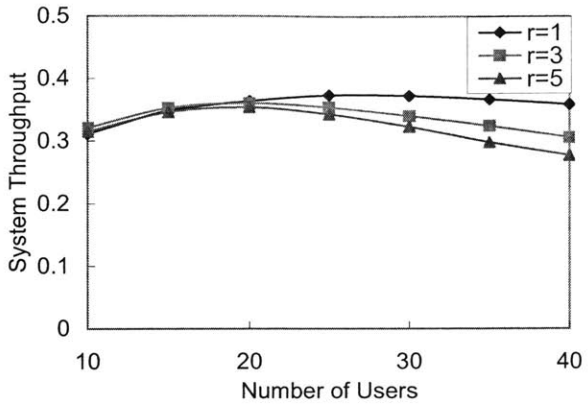


Figure 4-18: System Throughput vs Number of Users for Different $r$ - without Backoff ($D = 30$ and $p = 0.7$)

$p$, round trip time $D$, and number of users $N$, respectively. All the three figures show a throughput improvement by MAC layer retransmissions. This is consistent with our previous discussions, that is, due to the long duration of the TCP backoff periods, systems with TCP backoff always operate in regions with many idle slots and light load, and MAC layer retransmissions are preferred for such systems.

## 4.6 Conclusions

In this chapter we study the interaction between TCP and ALOHA by examining the system performance with TCP at the transport layer and ALOHA at the MAC layer. Two simple equations are derived from which the system performance can be solved directly, and the optimal MAC layer transmission probability at which the system achieves its highest throughput is given. The maximum possible system throughput is shown to be $1/e$, and a sufficient and necessary condition to achieve this throughput is derived, which cannot always be satisfied.

In addition, for systems with TCP timeout backoff, due to the long duration of backoff periods, the system always operates at a range with too many idle slots, and hence has a throughput below $1/e$. Whereas for systems without TCP timeout backoff, adjusting the MAC layer transmission probability can change the system load. In particular, when the system operates with too many collisions, such as systems with small round trip time and large number of users, the throughput of $1/e$ can always be achieved by lowering the MAC

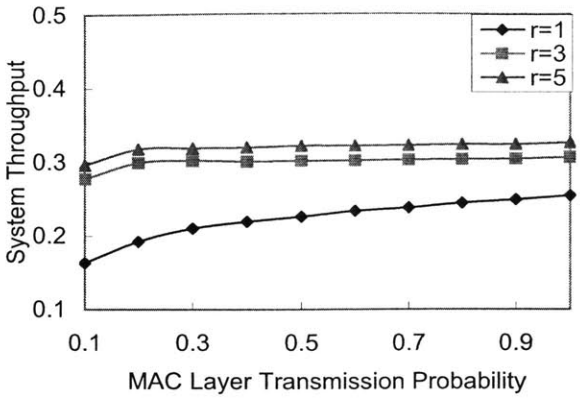


Figure 4-19: System Throughput vs Transmission Probability for Different $r$ - with Backoff ($N = 10$ and $D = 10$)

layer transmission probability. However, in systems with large propagation delay and small number of users, the throughput of $1/e$ is not always achievable, because even when the transmission probability is set to its maximum value of 1, the system remains under-loaded. Moreover, since MAC layer retransmissions react faster to collisions and hence inject more traffic into the channel, MAC layer retransmissions are preferred for systems with light load. In contrast, when the system is heavily loaded, retransmissions at the transport layer (TCP) result in higher throughput.

Overall, from the MAC layer perspective, the system performance is a result of the balance between idle slots and collisions, while from the perspective of the transport layer, the system performance is a tradeoff between TCP packet loss probability and round trip time.

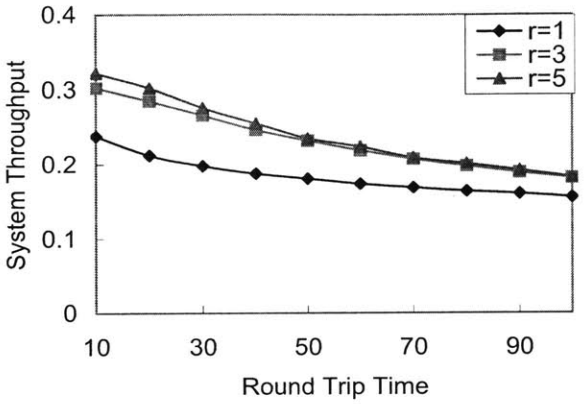


Figure 4-20: System Throughput vs Round Trip Time for Different $r$ - with Backoff ($N = 10$ and $p = 0.7$)

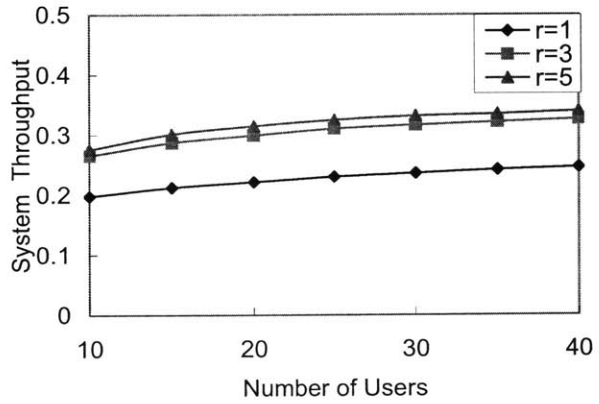


Figure 4-21: System Throughput vs Number of Users for Different $r$ - with Backoff ($D = 30$ and $p = 0.7$)

# Chapter 5

# Packet Scheduling with Window Service Constraints

## 5.1 Introduction

Traditionally, the performance of scheduling policies is measured by the mean response time, defined to be the difference between the departure and arrival times of a job. Among all policies, for a work-conserving queue, the Shortest-Remaining-Processing-Time (SRPT) scheduling policy is optimal with respect to minimizing the mean response time [46, 48]. The author in [47] further gives the distribution of the response time for M/G/1 queue under the SRPT policy. Recently a number of papers [9, 56, 42] address the fairness property of the SRPT policy using slowdown (also called stretch), which is defined to be the ratio of the response time and the processing time of a job, as the measure. They show that the SRPT policy not only minimizes the mean response time, but also is good in fairness.

All these previous works assume that upon the arrival of a job, any part of the job is available for service. In practice, jobs are often broken into smaller units before being served, and there may exist a limit on the number of units that can be served within a time interval. For example, in data networks, files are broken into messages and then packets before being released from the transport layer. If the transport layer employs TCP, then the number of packets that can be released to the lower layer is limited by the current window size of TCP, denoted by $W$. That is, at most $W$ packets can be released within one round trip time. Another example is the transmission of frames at the data link layer,

where the number of frames that can be transmitted within one round trip time is limited by the window size of the data link layer protocol.

This chapter considers the optimal scheduling policy that minimizes the mean response time when there exists a limit on the number of units to be served within a fixed time interval. For convenience, we call the limit window size and the units packets. For simplicity, we consider the scheduling of packets from two jobs. The effects of window constraints are presented in detail, and an optimal policy is developed. Based on this optimal policy, a suboptimal policy which gives further insights on schedule is also derived. In both policies, both the job lengths and the window sizes are essential. Moreover, instead of changing priority of jobs at different times, in most cases the optimal policy gives full priority to one job.

The chapter is organized as follows: in the next section we provide the problem formulation and discuss the effects of window constraints. In Section 5.3 we describe the optimal policy and the suboptimal policy. In Section 5.4 we conclude the chapter.

## 5.2 Problem Description and Effects of Window Constraints

In this section we first describe the system considered, then explore the window constraints.

### 5.2.1 Problem Description

Consider a system with two jobs to be served by one server, as shown in Figure 5-1. The two jobs are broken into packets before being served. They have $L_1$ and $L_2$ packets, respectively. All packets have the same length. The server performs packets-based service. That is, a new packet cannot preempt the packet being served, but after the service is complete, the server can process packets from either job. The processing time of one packet is defined to be one time slot. In this way the system becomes a slotted system, and henceforth all time is measured in time slots.

For convenience, starting from time 0, we index the time slots in order. That is, the $k$th slot is called slot $k$. Similarly, we index the packets from job $i$, $i = 1, 2$, in order, too. That is, the $p$th packet served from job $i$ is called packet $p$ of job $i$.

The service is under window constraints. By under window constraints, we mean that within a fixed time interval $\tau$ slots, the server can serve at most $W_1$ packets from job 1 and

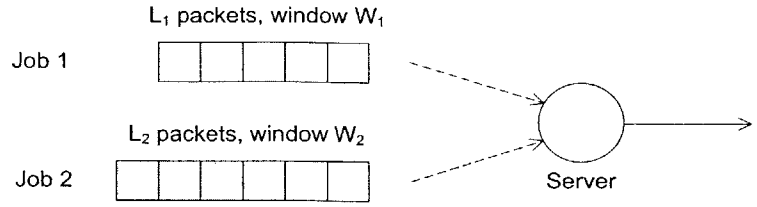


Figure 5-1: System with Two Jobs and One Server

$W_2$ packets from job 2 , where $W_i$, $i = 1, 2$, is called the window size of job $i$. The interval $\tau$ corresponds to the round trip time in data networks, and the window size $W_i$ corresponds to the maximum number of outstanding packets allowed in data networks.

For an arbitrary scheduling policy $\pi$, let $t_i(\pi)$, $i = 1, 2$, be the time slot that job $i$ is finished and define cost $c(\pi) = t_1(\pi) + t_2(\pi)$. We want to find a scheduling policy that minimizes $c(\pi)$, that is, minimizes the average response time of the two jobs. For brevity, later on when there is no ambiguity, we omit $\pi$ and write $c$, $t_1$ and $t_2$ directly.

### 5.2.2 Equivalent Constraints

From the above problem description we can see that if there were no window constraints, the problem would be the traditional scheduling problem that minimizes the average response time, and the SRPT policy is the optimal solution. The essence of the window constraints is to limit the availability of packets for service.

One equivalent constraint to the window constraints is on the availability of an arbitrary packet for service at a time slot, and we call it packet constraint. Specifically, let $u_i(p)$ denote the time slot that packet $p$ from job $i$ is served. At time 0, there are $W_1$ packets from job 1 (packet 1 to $W_1$) and $W_2$ packets from job 2 (packet 1 to $W_2$) that are available for service. For any other packet $p \geq W_i + 1$ from job $i$, $i = 1, 2$, the packet constraint says that at an arbitrary slot $z$, packet $p$ is available for service if and only if

$$u_i(p - W_i) \leq z - \tau. \tag{5.1}$$

This packet constraint is helpful in drawing the service pattern under a scheduling policy with the window constraints.

Another equivalent constraint is on the difference between the slots when packet $p$ and

packet $p - W_i$ receive service, and we call it service constraint. This constraint is useful in deriving the optimal policy. Specifically, the service constraint requires

$$u_i(p) - u_i(p - W_i) \geq \tau \text{ for all } p \geq W_i + 1. \tag{5.2}$$

That is, for packets that are $W_i$ packets apart, the slots when they receive service must be at least $\tau$ apart. The equality holds if and only if packet $p$ is served at the time slot when it becomes available. Notice that since each packet takes one time slot for processing, $u_i(p) - u_i(p - W_i) \geq W_i$ holds for all possible policies. Also notice that by definition, $t_i = u_i(L_i)$.

### 5.2.3 Service Pattern under Window Constraints - One Job Case

In this subsection we illustrate how the window constraints affect the service pattern when there is only one job, say job $i$, waiting for service. Consider the work-conserving policy. By work-conserving, we mean that no work is created or destroyed in the system, therefore the server cannot be idle if there are packets available for service. If there were no window constraints, then the server would continue serving packets until the job is finished. When there exists window constraint, by repeatedly using the packet constraint (Equation (5.1)), it is straightforward to obtain the service pattern under the work-conserving policy, as shown in Figure 5-2 (a) and (b) for case $W_i \geq \tau$ and case $W_i < \tau$, respectively. The gray blocks in the figure represent that the server is processing packets, and the letter inside each block represents which job the packets in service are from. The formula above the blocks are the number of packets in the packet-blocks or the lengths of the idle-blocks in slots, and the formula below are the block lengths in slots. Later figures showing service patterns can be explained similarly.

Figure 5-2 (a) shows that when $W_i \geq \tau$, the service pattern is the same as that without window constraints. That is, the server continues serving packets until the job is finished. Actually, since the server can serve at most $\tau$ packets during any $\tau$ interval, which cannot be greater than the window size $W_i$ in this case, the window constraints are always satisfied and hence take no effects in practice. Another interpretation is that in this case, $u_i(p) - u_i(p - W_i) \geq W_i \geq \tau$ for all $p \geq W_i + 1$. That is, the service constraints (5.2) are automatically. Therefore, when $W_i \geq \tau$, there are always packets available for service,

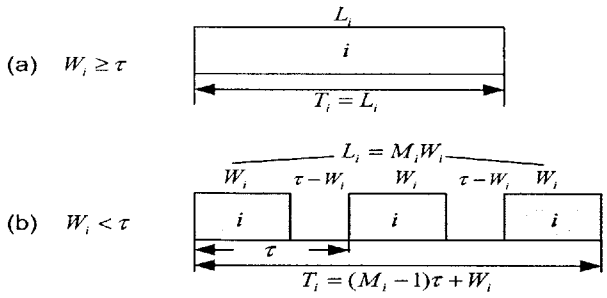


Figure 5-2: Service Pattern for One Job under Work-Conserving Policy and Window Constraints

and the service pattern is the same as that without window constraints. Whereas when $W_i < \tau$, the service constraint is not satisfied automatically, and not all time slots have packets available for service. The service pattern thus consists of alternative packet-blocks and idle-blocks. Figure 5-2 (b) shows that their sizes are $W_i$ and $\tau - W_i$, respectively.

Moreover, for convenience, define function $f_i(l)$ to be the shortest time needed to finish $l$ packets from job $i$ when there is no packets from other jobs. When $W_i \geq \tau$, it is obvious that $f_i(l) = l$. When $W_i < \tau$, further define $m$ and $n$ to be the integers that satisfy $l = mW_i + n$ and $n \in [1, W_i]$. $f_i(l)$ can then be obtained by considering the slots when packet $l - kW_i$ receives service for $k = 0, .., m$. Specifically, by the service constraint (5.2), $u_i(l) - u_i(n) = \sum_{k=0}^{m-1}[u_i(l - kW_i) - u_i(l - (k+1)W_i)] \geq m\tau$. Recall that packet 1 to $W_i$ are available for service at time 0. Therefore $u_i(n) \geq n$ and consequently, $u_i(l) \geq m\tau + n$. Moreover, it is easy to verify that $u_i(l) = m\tau + n$ can be achieved by the work-conserving policy. We thus have that the shortest time $f_i(l) = m\tau + n$ for $W_i < \tau$. Overall,

$$f_i(l) = \{ \begin{array}{ll} l & W_i \geq \tau \\ m\tau + n & W_i < \tau \end{array} \tag{5.3}$$

Figure 5-3 plots function $f_i(l)$ for different window size $W_i$.

Further define $T_i$ to be the shortest time needed to finish job $i$ when there is no other jobs, and call it shortest processing time of job $i$ subject to window constraint. For simplicity, assume $L_i = M_i W_i$ for some integer $M_i$ when $W_i < \tau$. Then by definitions of $T_i$ and $f_i(l)$ and Equation (5.3),

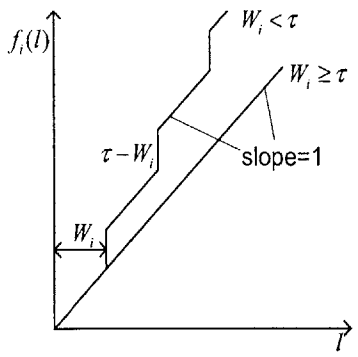


Figure 5-3: Time Needed for Serving $l$ Packets from Job $i$

$$T_i = f_i(L_i) = \begin{cases} L_i & \text{when } W_i \geq \tau \\ (M_i - 1)\tau + W_i & \text{when } W_i < \tau \end{cases} \tag{5.4}$$

This is consistent with the $T_i$'s shown in Figure 5-2. Note that for a job with length $L_i$ and a server with window constraints $W_i$ and $\tau$, $T_i$ is a constant and independent of the scheduling policy employed.

In summary, different window sizes lead to different packet service patterns. Specifically, when $W_i \geq \tau$, the window constraints are satisfied automatically and take no effects, and the server continues serving the job until it is finished. The service pattern is thus the same as that without window constraints and is one packet block. Whereas when $W_i < \tau$, the service pattern consists of alternative packet blocks and idle blocks, with length $W_i$ and $\tau - W_i$, respectively. Moreover, Equation (5.3) gives the shortest time needed to serve $l$ packets from job $i$, and Equation (5.4) gives the shortest time needed to serve job $i$. Note that $T_i$ is a property of job $i$ and is independent of the scheduling policy implemented.

### 5.2.4 Optimality of the SRPT Policy under Window Constraints

For traditional scheduling problems that minimizes the average response time, there are no window service constraints, and the SRPT policy is optimal. However, with window service constraints, the SRPT policy may no longer be optimal, as shown by the following example.

Consider the case when the fixed time interval $\tau = 4$, the job lengths of the two jobs are $L_1 = 4$ and $L_2 = 5$, respectively, and the window sizes are $W_1 = 2$ and $W_2 = 1$, respectively. Since $L_1 < L_2$, the SRPT policy gives priority job 1. By repeatedly using the

packet constraint (Equation (5.1)), Figure 5-4 (a) plots the service pattern under the SRPT policy. From the figure we can see that the cost of the SRPT policy is 25. Figure 5-4 (b) plots the service pattern under the policy that gives priority to job 2. Clearly, the cost of this policy is 24, which is less than the cost of the SRPT policy. Therefore, in this case, the SRPT policy is not optimal.

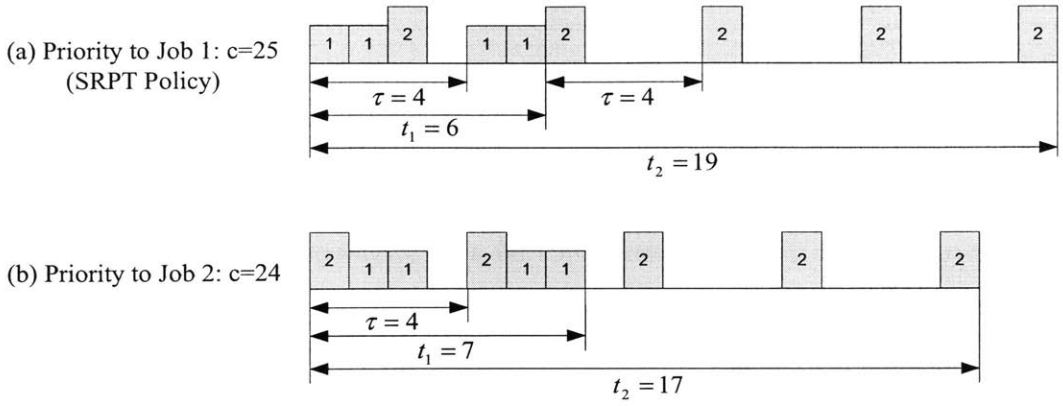


Figure 5-4: Illustration of SRPT Policy

Since window constraints introduce idle slots in service patterns, one conjecture is that policy with less idle slots has lower cost. This conjecture is not true either, as shown by the following example. Consider the case when the fixed time interval $\tau = 3$, the job lengths of the two jobs are $L_1 = 3$ and $L_2 = 6$, respectively, and the window sizes are $W_1 = 3$ and $W_2 = 2$, respectively. Figure 5-5 (a) and (b) plot the service patterns under policies that give priority to job 1 and job 2, respectively. Clearly, although the policy that gives priority to job 2 has no idle slots, its cost 17 is larger than the cost 14 of the policy that gives priority to job 1, which has two idle slots. Obviously in this case, policy that has less idle slots does not have lower cost.

It is hence of interest to understand with window constraints, which parameters are essential to the optimal policy, the job lengths, the window sizes, the number of idle slots, or a combination of them. Later after we obtain the optimal policy, we will see that both the job lengths and the window sizes are essential to the optimal policy.

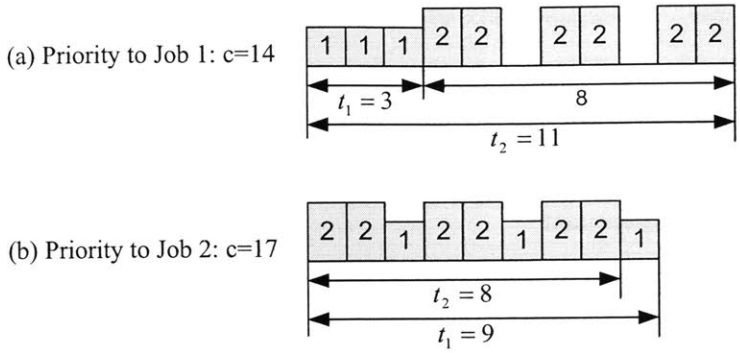


Figure 5-5: Illustration of Policy with Less Idle Slots

### 5.2.5 Service Pattern under Window Constraints - Two Job Case

Next consider how the window constraints affect the service pattern when there are two jobs. Start from the simple case when the server employs the work conserving policy that gives full priority to job 1. By giving full priority to job 1, we mean that whenever packets from job 1 are available for service, the server serves packets from job 1. Under this scheduling policy, the service pattern for job 1 is unaffected and thus the same as that when there is only job 1 waiting for service (c.f. Figure 5-2).

We classify different cases by whether $W_1 \geq \tau$, $W_1 + W_2 > \tau$ and/or $L_1 + L_2 > T_1$. The reasons will be explained later. The service patterns under different cases can be obtained straightforwardly by using the packet constraint, and the results are as shown in Figure 5-6 to Figure 5-9.

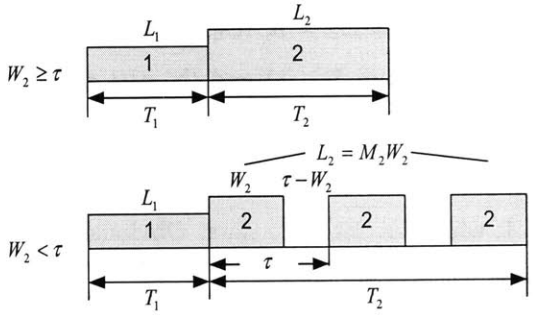


Figure 5-6: Service Pattern for Two Jobs under Work-Conserving Policy with Full Priority to Job 1, $W_1 \geq \tau$

Specifically, Figure 5-6 plots the case when $W_1 \geq \tau$. As mentioned before, in this case

there are always packets from job 1 available for service and the server continues serving job 1 until it is finished. Afterwards the server begins to serve job 2 in the absence of job 1. Therefore the service pattern can be divided into two parts: the first part is the service pattern for job 1 in the absence of job 2 (takes $T_1$ slots), and the second part is the service pattern for job 2 in the absence of job 1 (takes $T_2$ slots).

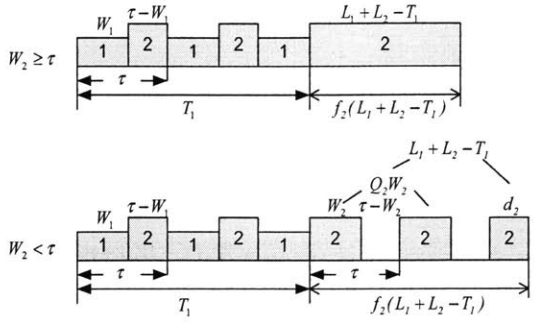


Figure 5-7: Service Pattern for Two Jobs under Work-Conserving Policy with Full Priority to Job 1, $W_1 < \tau$, $W_1 + W_2 > \tau$ and $L_1 + L_2 > T_1$

Figure 5-7 shows the service pattern when $W_1 < \tau$, $W_1 + W_2 > \tau$ and $L_1 + L_2 > T_1$. The inequality $W_1 < \tau$ means that packets from job 1 are not always available for service, and there are gaps between packet-blocks of job 1. Moreover, during any time interval with length $\tau$, the server can serve at most $\tau$ packets, while according to the window constraints, there can be as many as $W_1 + W_2 > \tau$ packets available for service before either job is finished. Therefore, at any time slot before either job is finished, there exists at least one packet, from either job 1 or job 2, available for service. Consequently, a work-conserving policy has no idle slot before either job is finished, and hence packets from job 2 fill up the gaps between the packet-blocks of job 1, as shown in Figure 5-7.

Furthermore, since each packet takes one time slot, $T_1 - L_1$ packets are needed to fill up all the gaps between packet-blocks of job 1. Therefore, inequality $L_1 + L_2 > T_1$ means that there are enough packets from job 2 to fill all the gaps between packet-blocks of job 1, and at the time job 1 is finished (slot $T_1$), there are $L_1 + L_2 - T_1$ packets left from job 2. The service pattern in Figure 5-7 can thus be summarized as follows: the pattern for job 1 is the same as that in Figure 5-2 (without job 2). Before job 1 is finished, packets from job 2 fill up all the gaps between packet-blocks of job 1, and there are no idle slots. After job 1 is finished, the service pattern follows the pattern of job 2 for its remaining packets.

For convenience, for this case, define $Q_2$ and $d_2$ to be the integers that satisfy

$$L_1 + L_2 - T_1 = Q_2 W_2 + d_2, \tag{5.5}$$

and $d_2 \in [1, W_2]$. The physical meaning of $Q_2$ and $d_2$ are shown in Figure 5-7. By symmetry, when $W_2 < \tau$, $W_1 + W_2 > \tau$ and $L_1 + L_2 > T_2$, define $Q_1$ and $d_1$ to be the integers that satisfy $d_1 \in [1, W_1]$ and

$$L_1 + L_2 - T_2 = Q_1 W_1 + d_1. \tag{5.6}$$

These parameters will be used later when we describe the optimal policy.

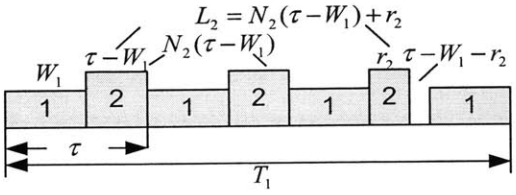


Figure 5-8: Service Pattern for Two Jobs under Work-Conserving Policy with Full Priority to Job 1, $W_1 < \tau$, $W_1 + W_2 > \tau$ and $L_1 + L_2 \leq T_1$

Figure 5-8 plots the service pattern when $W_1 < \tau$, $W_1 + W_2 > \tau$ but $L_1 + L_2 \leq T_1$. Similar to the previous case, packets from job 2 fill up the gaps between packet-blocks of job 1. Differently, in this case $L_1 + L_2 \leq T_1$. This means that there are no enough packets from job 2 to fill up all the gaps ($T_1 - L_1$ packets needed to fill up all the gaps), and job 2 is finished before job 1. For convenience, define $N_2$ and $r_2$ to be the integers that satisfy

$$L_2 = N_2(\tau - W_1) + r_2, \tag{5.7}$$

and $r_2 \in [1, \tau - W_1]$. Their physical meanings are shown in Figure 5-8. By symmetry, when $W_2 < \tau$, $W_1 + W_2 > \tau$ but $L_1 + L_2 \leq T_2$, define $N_1$ and $r_1$ to be the integers that satisfy $r_1 \in [1, \tau - W_2]$ and

$$L_1 = N_1(\tau - W_2) + r_1. \tag{5.8}$$

These parameters will be used later when we describe the optimal policy.

Finally, Figure 5-9 plots the service pattern when $W_1 + W_2 \leq \tau$. In this case, during

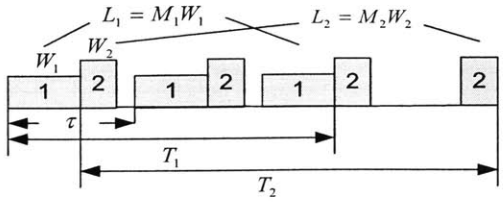


Figure 5-9: Service Pattern for Two Jobs under Work-Conserving Policy with Full Priority to Job 1, $W_1 + W_2 \leq \tau$

each $\tau$ time slots, there are at most $W_1 + W_2$ packets served and the rest $\tau - (W_1 + W_2)$ slots are idle.

The above analysis shows that under the work-conserving scheduling policy that gives full priority to job 1, the inequality $W_1 \geq \tau$ determines whether the server continues serving job 1 until it is finished, or there are gaps between packet-blocks of job 1. The inequality $W_1 + W_2 > \tau$ determines whether there are idle slots before either job is finished. Moreover, when $W_1 < \tau$ and $W_1 + W_2 > \tau$, the inequality $L_1 + L_2 > T_1$ determines whether job 1 is finished first. The service pattern for other policies under different window sizes and job lengths can be obtained in a similar way.

The above analysis and figures also show that due to the window constraints, the service pattern for jobs are quite different from traditional problems without window constraints. In the next section we will show that the optimal policy is quite different as well.

## 5.3 Optimal Policy and Suboptimal Policy

In this section we derive the optimal policy for the problem described and the more insightful suboptimal policy. The difference between the costs of the optimal policy and this suboptimal policy will be given as well. When deriving the optimal policy, we classify the problem into two cases:

Case 1: $W_1 + W_2 \leq \tau$;

Case 2: $W_1 + W_2 > \tau$.

In the case when $W_i < \tau$ and $M_i = 1$ ($L_i = W_i$), the window constraints take no effects. Since we are concerning only the window constraints, henceforth when $W_i < \tau$, we only

consider the cases with $M_i \geq 2$ .

### 5.3.1 Optimal Policy When $W_1 + W_2 \leq \tau$

**Theorem 5.1.** *If $W_1 + W_2 \leq \tau$, the policy $\pi^*$ that gives full priority to the job with smaller window size is optimal.*

*Proof.* Since $W_i > 0$ for $i = 1, 2$, the inequality $W_1 + W_2 \leq \tau$ means that $W_i < \tau$ for both $i = 1$ and 2. From the service constraint (5.2), we have $u_i(kW_i) - u_i((k-1)W_i) \geq \tau$ for $k \in [2, M_i]$, where as defined before, $M_i$ is the integer that satisfies $L_i = M_i W_i$. Equality holds if and only if packet $kW_i$ is served at the time slot when it becomes available. By summing this inequality over $k$, we obtain

$$\begin{aligned}(M_i - 1)\tau &\leq \sum_{k=2}^{M_i} [u_i(kW_i) - u_i((k-1)W_i)] \\ &= u_i(M_i W_i) - u_i(W_i) \\ &= t_i - u_i(W_i),\end{aligned}$$

where the last equality comes from the definitions of $t_i$ and $u_i$. We therefore have

$$c = t_1 + t_2 \geq (M_1 - 1)\tau + (M_2 - 1)\tau + u_1(W_1) + u_2(W_2). \tag{5.9}$$

Recall that packet 1 to $W_1$ from job 1 and packet 1 to $W_2$ from job 2 are available for service from time 0. Hence the problem of minimizing $u_1(W_1) + u_2(W_2)$, called the suboptimal problem, is the traditional optimization problem with no window constraints and the SRPT policy that gives priority to the job with shorter $W_i$ is optimal. Without loss of generality, assume $W_1 \leq W_2$. Then by applying the result of the SRPT policy, the resulting optimal sum of the suboptimal problem is $2W_1 + W_2$, which is a lower bound for $u_1(W_1) + u_2(W_2)$. After plugging it into the inequality (5.9), we obtain

$$\begin{aligned}c &\geq (M_1 - 1)\tau + (M_2 - 1)\tau + 2W_1 + W_2 \\ &= T_1 + T_2 + W_1,\end{aligned}$$

where the equality follows from Equation (5.4). The above inequality gives us a lower bound for $c$ for any policy that satisfies the window constraints.

Now consider the policy that gives full priority to the job with the shorter window size, which is job 1 under our assumption. From Figure 5-9 we can easily obtain that under this scheduling policy, $c = t_1 + t_2 = T_1 + T_2 + W_1$, which achieves the lower bound of $c$. Therefore, the policy that gives full priority to the job with the shorter window size is optimal.

□

From the proof we see that the lower bound $T_1 + T_2 + W_1$ can be achieved if and only if the equality in (5.9) is achieved and $u_1(W_1) + u_2(W_2)$ achieves its lower bound. It can be easily verified that our optimal policy meets both of the two requirements.

### 5.3.2 Optimal Policy When $W_1 + W_2 > \tau$

In this subsection we consider the second case, when $W_1 + W_2 > \tau$. We first give five lemmas that characterize the optimal policy, then based on these lemmas, we develop the optimal policy.

**Lemma 5.1.** *If $W_1 + W_2 > \tau$, there exists an optimal policy that has no idle slots before either job is finished.*

*Proof.* Suppose policy $\pi$ is an optimal policy and has at least one idle slot before either job is finished. We now construct another policy $\pi'$ that has no idle slots before either job is finished and whose cost is at most $c(\pi)$, as follows. Since $W_1 + W_2 > \tau$, at each of those idle slots under policy $\pi$, there is at least one packet from job 1 or job 2 available for service. Serve one of these available packets at these idle slots. Repeat the process until either job 1 or job 2 is finished. Clearly, from the construction process, policy $\pi'$ has no idle slots before either job is finished. Moreover, $t_1(\pi') \leq t_1(\pi)$ and $t_2(\pi') \leq t_2(\pi)$, which results in $c(\pi') = t_1(\pi') + t_2(\pi') \leq t_1(\pi) + t_2(\pi) = c(\pi)$. Recall that policy $\pi$ is an optimal policy. Hence, policy $\pi'$ is also an optimal policy, but with no idle slots before either job is finished.

□

Define policy set $\Pi_i = \{\text{policy } \pi | \text{ job } i \text{ is finished first and there is no idle slot before job } i \text{ is finished}\}$, $i = 1, 2$, and optimal policy set $\Pi^* = \{\text{policy that is optimal and has no idle slots before either job is finished}\}$. The above lemma shows that $\Pi^*$ is not empty and $\Pi^* \subseteq \Pi_1 \cup \Pi_2$.

**Lemma 5.2.** *If $W_1 + W_2 > \tau$, then $L_1 + L_2 > \min\{T_1, T_2\}$. Furthermore, if $L_1 + L_2 \leq T_1$, then $\Pi_1 = \varnothing$ and $\Pi^* \subseteq \Pi_2$; if $L_1 + L_2 \leq T_2$, then $\Pi_2 = \varnothing$ and $\Pi^* \subseteq \Pi_1$.*

*Proof.* First show that $L_1 + L_2 > \min\{T_1, T_2\}$. If at least one of $W_1$ and $W_2$ is greater than or equal to $\tau$, say $W_i \geq \tau$, for $i = 1$ and/or 2, then $T_i = L_i$ (Equation (5.4)). Thus $L_1 + L_2 > T_i \geq \min\{T_1, T_2\}$. If $W_1 < \tau$ and $W_2 < \tau$, we show $L_1 + L_2 > \min\{T_1, T_2\}$ by contradiction, as follows.

Suppose $L_1 + L_2 \leq T_1$ and $L_1 + L_2 \leq T_2$. If $M_1 - 1 \leq M_2$, the first inequality $L_1 + L_2 \leq T_1$ gives us $M_2 W_2 = L_2 \leq T_1 - L_1 = (M_1 - 1)(\tau - W_1) \leq M_2(\tau - W_1)$, which results in $W_1 + W_2 \leq \tau$. Contradiction.

Similarly, if $M_1 - 1 > M_2$, the second inequality $L_1 + L_2 \leq T_2$ gives us $M_1 W_1 = L_1 \leq T_2 - L_2 = (M_2 - 1)(\tau - W_2) < M_1(\tau - W_2)$, which results in $W_1 + W_2 < \tau$. Contradiction.

Therefore when $W_1 + W_2 > \tau$, the two inequalities $L_1 + L_2 \leq T_1$ and $L_1 + L_2 \leq T_2$ cannot hold together. Hence, $L_1 + L_2 > \min\{T_1, T_2\}$.

Next consider the rest of the lemma. Since $T_1$ is the shortest processing time of job 1 subject to window constraint, for any policy $\pi$, the time when job 1 is finished $t_1 \geq T_1$. If $L_1 + L_2 \leq T_1$, then all policies that finish job 1 first have at least one idle slot among the first $T_1$ slots, thus among the first $t_1$ slots, that is, before job 1 is finished. Therefore by the definition of $\Pi_1$ and Lemma 5.1, $\Pi_1 = \varnothing$ and $\Pi^* \subseteq \Pi_2$.

Similarly, when $L_1 + L_2 \leq T_2$, $\Pi_2 = \varnothing$ and $\pi^* \subseteq \Pi_1$. □

The next three lemmas characterize policies in $\Pi_2$ which leads to optimal policies. By symmetry, similar results can be obtained for $\Pi_1$. Here we consider set $\Pi_2$ instead of $\Pi_1$ in order to use Figure 5-2 to 5-9 as illustration.

For an arbitrary policy $\pi_2 \in \Pi_2$, let $v_1(\pi_2)$ denote the number of packets from job 1 that were served before job 2 is finished. For clarity, later on when there is no ambiguity, we omit $\pi_2$ and use $v_1$ directly.

**Lemma 5.3.** *When $W_1 + W_2 > \tau$ and $L_1 + L_2 > T_2$, for any policy $\pi_2 \in \Pi_2$, we have $v_1(\pi_2) \in [v_1^L, v_1^U]$, where*

$$v_1^L = \begin{cases} 0 & \text{when } W_2 \geq \tau \\ T_2 - L_2 & \text{when } W_2 < \tau \end{cases} \tag{5.10}$$

$$v_1^U = \begin{cases} L_1 - 1 & when\ L_1 + L_2 > T_1 \\ (N_2+1)W_1 & when\ L_1 + L_2 \leq T_1 \end{cases} \tag{5.11}$$

*Proof.* First consider the lower bounds. A trivial lower bound is 0. For the case when $W_2 < \tau$, the tighter lower bound given in the lemma can be shown by contradiction as follows. Suppose $v_1 < T_2 - L_2$. Then the total number of packets served before job 2 is finished is $v_1 + L_2 < T_2 \leq t_2$, where the last inequality come from the definition of $T_2$. Therefore, there are idle slots before job 2 is finished, which contradicts to the definition of $\Pi_2$.

Next consider the upper bound. A trivial upper bound is $L_1 - 1$, otherwise job 1 would be finished first. For the case when $L_1+L_2 \leq T_1$, we show $v_1 \leq (N_2+1)W_1$ by contradiction. Suppose $v_1 > (N_2+1)W_1$ and let slot $s$ denote the slot where the first $v_1$ packets from job 1 are finished. On the one hand, since for every $\tau$ slots, the server can serve at most $W_1$ packets from job 1, before slot $s$ there are at least $(N_2+1)(\tau - W_1)$ slots where the server is not serving packets from job 1. On the other hand, by the definition $v_1$, before slot $s$ there are at most $L_2 - 1$ packets from job 2 that were served. However, Equation (5.7) shows that $L_2 - 1 = N_2(\tau - W_1) + r_2 - 1 \leq (N_2+1)(\tau - W_1) - 1$. Therefore, before slot $s$, thus before $t_2$, there are at least one idle slot, which contradicts to the definition of $\Pi_2$. □

Intuitively, $v_1^L$ corresponds to $v_1$ under the policy that gives full priority to job 2, and $v_1^U$ corresponds to $v_1$ under the policy that gives full priority to job 1. Then Equation (5.10) and (5.11) can be easily obtained from Figure 5-2 to 5-8.

**Lemma 5.4.** *When $W_1 + W_2 > \tau$ and $L_1 + L_2 > T_2$, for any policy in $\Pi_2$ with $v_1 = v \in [v_1^L, v_1^U]$, the finish time $t_1$ and $t_2$ and the cost $c$ are lower bounded by*

$$t_2^L(v) = L_2 + v, \tag{5.12}$$

$$t_1^L(v) = t_2^L(v) + f_1(L_1 - v) + b, \tag{5.13}$$

$$c^L(v) = 2L_2 + 2v + f_1(L_1 - v) + b, \tag{5.14}$$

*respectively, where $b = \tau - W_1 - r_2$ if $L_1 + L_2 \leq T_1$ and $v = v_1^U = (N_2+1)W_1$, and $b = 0$ otherwise. Moreover, there exists at least one policy in $\Pi_2$ that achieves these bounds.*

*Proof.* Here we only prove the lower bounds. One policy that achieves both bounds can be constructed. Since the construction is complicated and provides no insights, plus skipping it does not affect the following part of the chapter, for brevity, we omit the details.

The lower bound $t_2^L(v)$ comes directly from the definition of $v_1$. For $t_1^L(v)$, by the definition of $v_1$, there are $L_1 - v$ packets from job 1 left upon the slot job 2 is finished, which need at least $f_1(L_1 - v)$ slots to be served. Therefore, $t_1 \geq t_2 + f_1(L_1 - v) \geq t_2^L(v) + f_1(L_1 - v)$.

A tighter lower bound can be found when $L_1 + L_2 \leq T_1$ and $v = v_1^U = (N_2 + 1)W_1$, as follows. Since $v = (N_2 + 1)W_1$, we have $L_1 - v = M_1 W_1 - (N_2 + 1)W_1 = (M_1 - N_2 - 1)W_1$. Hence from Equation (5.3), $f_1(L_1 - v) = (M_1 - N_2 - 2)\tau + W_1$. Therefore,

$$
\begin{aligned}
& t_2^L(v) + f_1(L_1 - v) + (\tau - W_1 - r_2) \\
=& (L_2 + v) + (M_1 - N_2 - 2)\tau + W_1 + (\tau - W_1 - r_2) \\
=& L_2 + (N_2 + 1)W_1 + (M_1 - N_2 - 2)\tau + W_1 + (\tau - W_1 - r_2) \\
=& (M_1 - 1)\tau + W_1 + L_2 - (N_2(\tau - W_1) + r_2) \\
=& T_1.
\end{aligned}
$$

where the last equality comes from Equation (5.4) and (5.7). By the definition of $T_1$, we have $t_1 \geq T_1$. Consequently, $t_1 \geq t_2^L(v) + f_1(L_1 - v) + (\tau - W_1 - r_2)$.

Combining the above results leads to the lower bound $t_1^L(v)$ given in the lemma.

The lower bound $c^L(v)$ for the cost follows from $c = t_1 + t_2 \geq t_1^L(v) + t_2^L(v) = c^L(v)$. □

Figure 5-8 illustrates the lower bound $t_1^L(v)$ in the lemma when $W_1 < \tau$ and $T_2 < L_1 + L_2 \leq T_1$.

Over the policy set $\Pi_2$, for a fixed $v_1 = v$, Lemma 5.4 gives the lowest cost. Lemma 5.3 further provides upper and lower bounds on $v_1$. We can thus optimize the lowest cost in Lemma 5.4 over all possible $v_1$ and obtain the optimal cost over $\Pi_2$. Specifically, for an arbitrary policy $\pi_2$ in $\Pi_2$, we have $c(\pi_2) \geq c^L(v_1(\pi_2)) \geq \min_{v \in [v_1^L, v_1^U]} c^L(v)$. Let $c_2^*$ denote the minimum and $v_1^*$ denote the $v$ that achieves the minimum. The next lemma develops this idea and gives $v_1^*$ over $\Pi_2$.

**Lemma 5.5.** *When $W_1 + W_2 > \tau$ and $L_1 + L_2 > T_2$, the lowest cost over $\Pi_2$, denoted by $c_2^*$, is achieved at the following $v_1^*$:*

1. *when* $W_1 \geq \tau$: $v_1^* = v_1^L$;

2. *when* $\tau/2 < W_1 < \tau$ *and* $W_2 \geq \tau$: $v_1^* = v_1^L = 0$;

3. *when* $\tau/2 < W_1 < \tau$ *and* $W_2 < \tau$:

   (a) *if* $Q_1 = 0$: $v_1^* = v_1^L = L_1 - d_1$;

   (b) *if* $Q_1 \neq 0$ *and* $d_1 \geq \tau - W_1$: $v_1^* = v_1^L = T_2 - L_2$;

   (c) *if* $Q_1 \neq 0$ *and* $d_1 < \tau - W_1$: $v_1^* = v_1^L + d_1 = (M_1 - Q_1)W_1$;

4. *when* $W_1 \leq \tau/2$ *and* $L_1 + L_2 > T_1$:

   (a) *if* $W_2 < \tau$ *and* $Q_1 = 0$: $v_1^* = v_1^L = L_1 - d_1$;

   (b) *if* $W_2 \geq \tau$ *or* $Q_1 \neq 0$: $v_1^* = (M_1 - 1)W_1$;

5. *when* $W_1 \leq \tau/2$ *and* $L_1 + L_2 \leq T_1$:

   (a) *if* $r_2 \geq W_1$: $v_1^* = v_1^U = (N_2 + 1)W_1$;

   (b) *if* $r_2 < W_1$: $v_1^* = v_1^U - W_1 = N_2 W_1$.

*Proof.* First consider part 1) when $W_1 \geq \tau$. In this case $f_1(l) = l$ and $L_1 + L_2 > T_1$. Hence from Lemma 5.4 we have: $c_2^L(v) = 2L_2 + 2v + L_1 - v = L_1 + 2L_2 + v$. Therefore, $c_2^L(v)$ increases monotonically with $v$ and consequently, $v_1^* = v_1^L$. Part 1) thus holds.

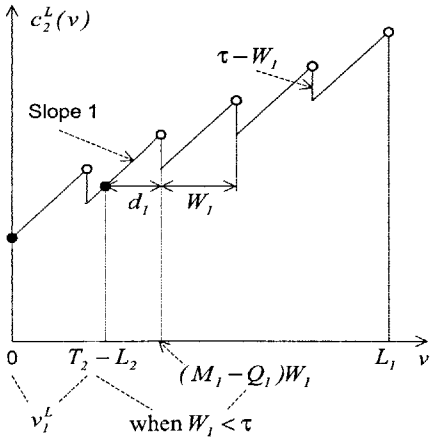


Figure 5-10: $c_2^L(v)$ against $v$ when $\tau/2 < W_1 < \tau$
(when $W_2 \geq \tau$, $v_1^L = 0$ and when $W_2 < \tau$, $v_1^L = T_2 - L_2$)

Next consider part 2) and 3) when $\tau/2 < W_1 < \tau$. From Lemma 5.4, Figure 5-10 plots $c_2^L(v)$ against $v$ for this case. This figure directly gives $v_1^*$ under different cases in part 2) and 3) of the lemma. Here for brevity, we omit the details.

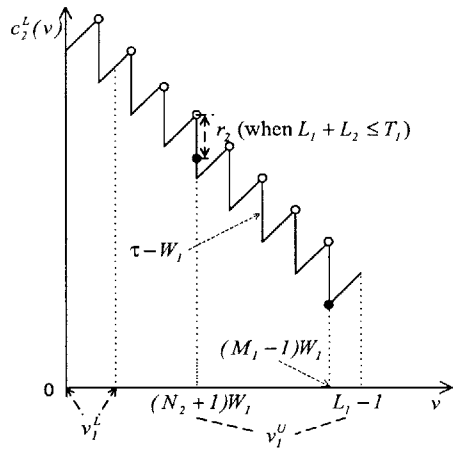


Figure 5-11: $c_2^L(v)$ against $v$ when $W_1 \le \tau/2$
(when $L_1 + L_2 > T_1$, $v_1^U = L_1 - 1$ and when $L_1 + L_2 \le T_1$, $v_1^U = (N_2 + 1)W_1$)

Now consider part 4) and 5) when $W_1 \le \tau/2$. From Lemma 5.4, Figure 5-11 plots $c_2^L(v)$ against $v$ for this case. The figure shows that when $L_1 + L_2 > T_1$, if $W_2 < \tau$ and $Q_1 = 0$, then $v$ lies in the last segment of the curve. Since along the last segment, $c_2^L(v)$ increases monotonically with $v$, we have in this case $v_1^* = v_1^L = L_1 - d_1$. This is part 4a). Whereas when $L_1 + L_2 > T_1$ but $W_2 \ge \tau$ or $Q_1 \ne 0$, the figure shows that $c_2^L(v)$ achieves its minimum at $v_1^* = (M_1 - 1)W_1$. This is part 4b).

Furthermore, when $W_1 \le \tau/2$ and $L_1 + L_2 \le T_1$, Lemma 5.3 shows that $v_1^U - v_1^L > W_1$. Figure 5-11 then gives $v_1^*$ as in part 5) of the lemma. Again for brevity, we omit the details. □

When $W_1 + W_2 > \tau$ and $T_2 < L_1 + L_2 \le T_1$, Lemma 5.2 says that $\Pi^* \subseteq \Pi_2$. That is, optimal policies over $\Pi_2$ are optimal policies over the entire policy space as well. Therefore, from Lemma 5.5, a policy $\pi \in \Pi_2$ with $v_1(\pi) = v_1^*$ is optimal over the entire policy space. We hence have the following theorems that give the optimal policy under different cases.

**Theorem 5.2.** *When $W_1 + W_2 > \tau$ and $T_2 < L_1 + L_2 \le T_1$, the following policy is optimal:*

1. *when $W_1 \le \tau/2$: if $r_2 \ge W_1$, give full priority to job 1; otherwise, serve $r_2$ packets from job 2 first, then give full priority to job 1.*

2. *when* $W_1 > \tau/2$*: if* $W_1, W_2 < \tau$ *and* $d_1 < \tau - W_1$*, serve* $d_1$ *packets from job 1 first, then give full priority to job 2; otherwise, give full priority to job 2.*

It is straightforward to verify that the policy given in the theorem has $v_1 = v_1^*$ defined in Lemma 5.5. Hence it is optimal. Figure 5-12 illustrates the policy for the four cases in the theorem.

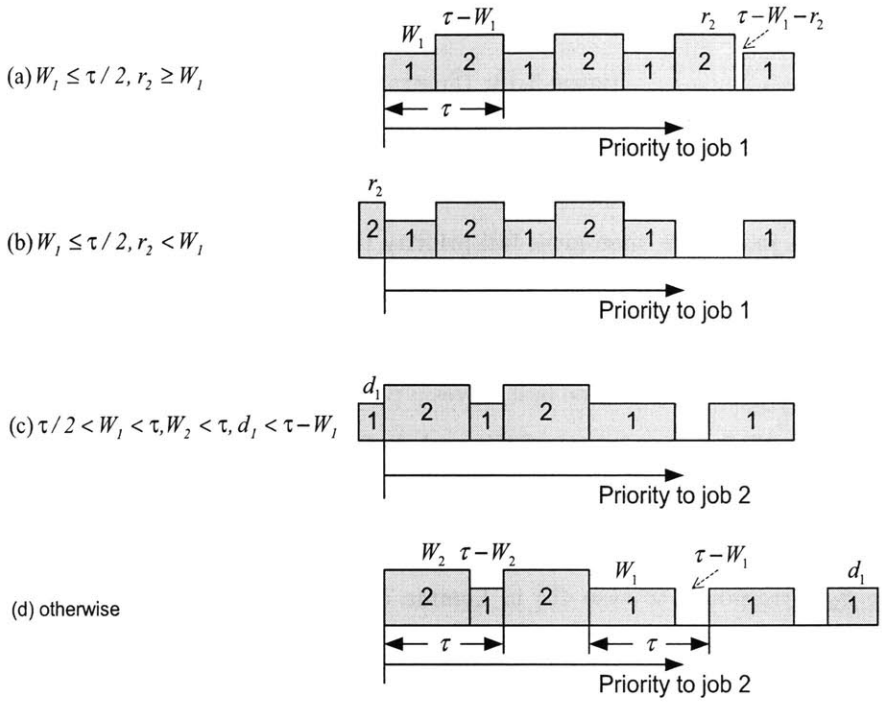


Figure 5-12: Illustration of Optimal Policy when $W_1 + W_2 > \tau$ and $T_2 < L_1 + L_2 \le T_1$

By symmetry, the next theorem follows, which gives optimal policy when $W_1 + W_2 > \tau$ and $T_1 < L_1 + L_2 \le T_2$ .

**Theorem 5.3.** *When* $W_1 + W_2 > \tau$ *and* $T_1 < L_1 + L_2 \le T_2$*, the following policy is optimal:*

1. *when* $W_2 \le \tau/2$*: if* $r_1 \ge W_2$*, give full priority to job 2; otherwise, serve* $r_1$ *packets from job 1 first, then give full priority to job 2.*

2. *when* $W_2 > \tau/2$*: if* $W_1, W_2 < \tau$ *and* $d_2 < \tau - W_2$*, serve* $d_2$ *packets from job 2 first, then give full priority to job 1; otherwise, give full priority to job 1.*

Theorem 5.2 and 5.3 provide optimal policies under case $T_2 < L_1 + L_2 \leq T_1$ and $T_1 < L_1 + L_2 \leq T_2$. We now consider the optimal policy for the last case, the case with $L_1 + L_2 > \max\{T_1, T_2\}$.

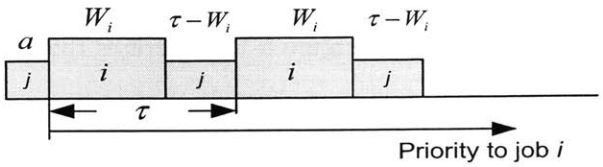


Figure 5-13: Illustration of Policy $\pi_{a,i}$

For $i, j = 1$ or 2, $i \neq j$ and an integer $a \leq W_i$, let $\pi_{a,i}$ denote the policy that serves $a$ packets from job $j$ first, then gives full priority to job $i$. Figure 5-13 illustrates this policy. In addition, when $W_1 < \tau/2$, $W_1 + W_2 > \tau$ and $L_1 + L_2 > T_1$, let $\hat{\pi}_2$ denote the following policy: if $W_2 \geq \tau$, then first gives priority to job 1 until $W_1$ packets from job 1 left, then gives priority to job 2; if $W_2 < \tau$, then first serves $n$ packets from job 1, next serves $m$ blocks of $W_2$ packets from job 2 and $m$ blocks of $W_1$ packets from job 1 alternatively, then gives priority to job 2. Here $m$ and $n$ are integers that satisfy $L_1 + L_2 - T_2 - W_1 = m(W_1 + W_2 - \tau) + n$ and $n \in [0, W_1 + W_2 - \tau]$. This policy is illustrated in Figure 5-14. It can be seen that policy $\hat{\pi}_2$ corresponds to case 4b) in Lemma 5.5 with $v_1(\hat{\pi}_2) = v_1^*$. Policy $\underline{\pi_1}$ is similarly defined.

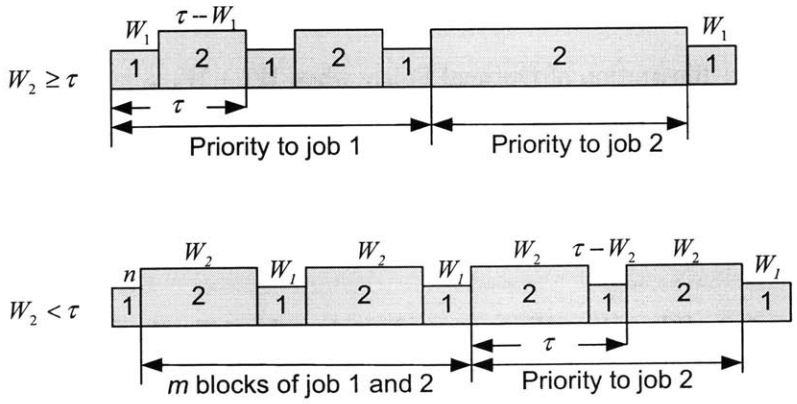


Figure 5-14: Illustration of Policy $\hat{\pi}_2$
(when $W_1 < \tau/2$, $W_1 + W_2 > \tau$ and $L_1 + L_2 > T_1$)

Next lemma gives the costs of the above policies.

Table 5.1: Optimal Policies over $\Pi_1$ or $\Pi_2$ when $W_1 + W_2 > \tau$ and $L_1 + L_2 > \max\{T_1, T_2\}$

| $W_1$ | $W_2$ | | |
|---|---|---|---|
| | $\geq \tau$ | $[\tau/2, \tau]$ | $\leq \tau/2$ |
| $\geq \tau$ | $\pi_{0,2}$,$\pi_{0,1}$ | $\pi_{0,2}$,$\pi_{0,1}$ | $\pi_{0,2}$,$\underline{\pi_1}$ |
| $[\tau/2, \tau]$ | $\pi_{0,2}$,$\pi_{0,1}$ | $\pi_{0,2}$,$\pi_{d_1,2}$,$\pi_{0,1}$,$\pi_{d_2,1}$ | $\pi_{0,2}$,$\pi_{d_1,2}$,$\pi_{0,1}$,$\underline{\pi_1}$ |
| $\leq \tau/2$ | $\hat{\pi}_2$,$\pi_{0,1}$ | $\pi_{0,2}$,$\hat{\pi}_2$,$\pi_{0,1}$,$\pi_{d_2,1}$ | N/A |

**Lemma 5.6.** *When $W_1 + W_2 > \tau$, for $i, j = 1$ or 2 and $i \neq j$, the cost of policy $\pi_{0,i}$ is*

$$c(\pi_{0,i}) = \begin{cases} 2T_i + f_j(L_1 + L_2 - T_i) & \text{when } L_1 + L_2 > T_i \\ T_i + (N_j + 1)\tau - (\tau - W_i - r_j) & \text{when } L_1 + L_2 \leq T_i \end{cases} \tag{5.15}$$

*When $W_1, W_2 < \tau$ and $L_1 + L_2 > T_i$, the cost of policy $\pi_{d_j,i}$ is*

$$c(\pi_{d_j,i}) = 2T_i + 2d_j + \begin{cases} f_j(L_1 + L_2 - T_i - d_j) & \text{when } Q_j \geq 1 \\ -W_i & \text{when } Q_j = 0 \end{cases} \tag{5.16}$$

*Moreover, when $W_i < \tau$ and $L_1 + L_2 \leq T_i$, the cost function of policy $\pi_{r_j,i}$ is $c(\pi_{r_j,i}) = T_i + N_j\tau + 2r_j$. When $W_j < \tau$ and $L_1 + L_2 > \max\{T_1, T_2\}$, the cost function of policy $\underline{\pi_i}$ is $c(\underline{\pi_i}) = 2(L_1 + L_2) - W_j$.*

These cost functions can be directly verified from Figure 5-2 to Figure 5-8.

Lemma 5.5 shows that when $L_1 + L_2 > \max\{T_1, T_2\}$, over policy set $\Pi_2$, $v_1(\pi_{0,2})$, $v_1(\pi_{d_1,2})$ or $v_1(\hat{\pi}_2)$ equals to $v_1^*$ for different cases. Therefore, policy $\pi_{0,2}$, $\pi_{d_1,2}$ or $\hat{\pi}_2$ is optimal for the corresponding cases over $\Pi_2$. By symmetry, policy $\pi_{0,1}$, $\pi_{d_2,1}$ or $\underline{\pi_1}$ is optimal for the corresponding cases over $\Pi_1$. Table 5.1 lists the optimal policies over $\Pi_1$ or $\Pi_2$ for different cases. For example, when $W_1, W_2 \geq \tau$, $\pi_{0,2}$ is optimal over $\Pi_2$ and $\pi_{0,1}$ is optimal over $\Pi_1$. We therefore have policy $\pi_{0,2}$ and $\pi_{0,1}$ in the cell for $W_1 \geq \tau$ and $W_2 \geq \tau$.

Furthermore, since $\Pi^* \subseteq \Pi_1 \cup \Pi_2$, one of the optimal policies over $\Pi_1$ and $\Pi_2$ is also optimal over the entire policy space. Although we can further classify different cases and derive their corresponding optimal policies, the number of categories is significant, and the description of optimal policies is tedious and provides little insights. We therefore do not further classify the cases, but rather give the following theorem:

**Theorem 5.4.** *When $W_1 + W_2 > \tau$ and $L_1 + L_2 > \max\{T_1, T_2\}$, for different cases in Table 5.1, the policy that has the lowest cost among policies in the corresponding cell is optimal.*

Note that the cost of policies listed are given in Lemma 5.6.

Theorem 5.2 to 5.4 provide optimal policies under different cases. They show that the optimal policy is a function of job lengths, window sizes as well as the time interval $\tau$. Furthermore, in most cases, the optimal policy does not change the priority of jobs but rather gives full priority to one job.

**Corollary 5.1.** *If $W_1, W_2 \geq \tau$, the policy that gives priority to the shorter job is optimal.*

*Proof.* When $W_i \geq \tau$ for some $i$, we have $T_i = L_i$, which leads to $L_1 + L_2 > T_i$. Therefore, when $W_1, W_2 \geq \tau$, $L_1 + L_2 > \max\{T_1, T_2\}$. The case thus falls into the first cell in Table 5.1, which lists policy $\pi_{0,2}$ and $\pi_{0,1}$. From Lemma 5.6, their costs are: $c(\pi_{0,2}) = 2T_2 + f_1 L_1 + L_2 - T_2 = 2L_2 + (L_1 + L_2 - L_2) = L_1 + 2L_2$, where the second equality comes from the fact $f_1(l) = l$ and $T_2 = L_2$. Similarly, $c(\pi_{0,1}) = 2L_1 + L_2$. Hence, if $L_1 \leq L_2$, then policy $\pi_{0,1}$ is optimal, and if $L_2 \leq L_1$, then policy $\pi_{0,2}$ is optimal. The corollary thus holds. □

As discussed before, when $W_1, W_2 \geq \tau$, the window constraints take no effect on the service of both jobs. The problem thus becomes the traditional problem that minimizes average response time. The optimal SRPT policy also gives priority to the shorter file in our case. This is consistent with the above corollary.

Although the policies described in Theorem 5.2 to 5.4 are optimal, they are complex and it is not clear which parameter is essential. We hence give a suboptimal policy in the following corollary, which gives more insights.

**Corollary 5.2.** *When $W_1 + W_2 > \tau$, the following policy is suboptimal: if $W_i \leq \tau/2$ for $i = 1$ or 2, give full priority to job $i$; if $W_i > \tau/2$ for both $i = 1$ and 2, give priority to the shorter job. The difference between the optimal cost and the cost of this suboptimal policy is less than* $\max\{W_1, W_2\}$.

This corollary can be proven by comparing the cost of the suboptimal policy described with the cost of the optimal policy described in Theorem 5.2 to 5.4, where the cost functions are given in Lemma 5.6. The comparison is tedious with no more information. We hence omit the details.

Notice that when $W_1 + W_2 > \tau$, the inequality $W_i \leq \tau/2$ for $i = 1$ or 2 means that job $i$ has smaller window size. We can thus combine Theorem 5.1 and Corollary 5.2 and obtain the following suboptimal policy for all cases:

**Theorem 5.5.** *The following policy is suboptimal for minimizing average response time of two jobs subject to window constraints: if both window sizes are greater than one half of the time interval, give full priority to the shorter job. Otherwise, give full priority to the job with the smaller window size. The difference between the optimal cost and the cost of this suboptimal policy is less than the maximum of the two window sizes.*

Now it's clear that when $W_i > \tau/2$ for both $i = 1$ and 2, the job lengths determine the optimal policy. Otherwise, the window sizes are essential.

## 5.4 Conclusions

This chapter considers the scheduling problem that minimizes the average response time of two jobs subject to window constraints. The effects of the window constraints are presented, and the optimal policy is derived. The results show that in most cases, instead of changing priority of jobs at different times, the optimal policy gives full priority to one job. In addition, in traditional optimization problems without window constraints, the remaining processing times are the only parameters that affects the optimal policy (SRPT policy). Whereas the suboptimal policy derived here shows that under window constraints, not only the job lengths (which correspond to the remaining processing times in traditional optimization problems) determine the optimal scheduling policy, but the relative magnitudes of the window sizes are essential as well.

One natural extension of the current work is to consider optimal scheduling problem with more than two jobs, and problems with random arrivals and departures. Another work of interest is to consider time-varying window sizes. From the application perspective, these extension work is expected to be helpful to understand the interaction between multiple TCP sessions and lower layer scheduling policy.

# Chapter 6

# Conclusions

By observing that cross-layer protocol interactions in heterogenous data networks may significantly degrade the system performance, in this thesis we investigate the interaction between TCP at the transport layer and ARQ at the data link layer, as well as the interaction between TCP and ALOHA at the MAC layer. We also study the scheduling problem with window service constraints, such as the link layer scheduling subject to TCP window constraints.

First we provide an analytical framework to study the interaction between transport layer TCP and data link layer ARQ. The system considered is a network that has high bandwidth-delay product and high link loss probability. The end nodes implement TCP at their transport layer and the error prone bottleneck link implements ARQ protocols. We model the system as a Markov chain with reward functions, and develop queuing models for both GBN and SRP, where the feedback delay is taken into account. The system throughput is then expressed in terms of system and protocol parameters. The analysis shows that in most cases implementing ARQ can achieve significant improvement in system throughput. In addition, the impacts of various protocol parameters on the system performance are also examined, such as the transport layer time-out value, the link layer packet size and the number of transmission attempts per packet at the link layer. We show that by proper setting of these parameters, system throughput can be improved by nearly an order of magnitude.

We then investigate the interaction between TCP and ALOHA by studying the system performance with TCP at the transport layer and ALOHA at the MAC layer. Two simple

equations are derived from which the system performance, such as the system throughput, can be obtained directly, and the optimal MAC layer transmission probability at which the system achieves its highest throughput is given. The analysis shows that the system performance is a result of the balance between idle slots and collisions. The maximum possible system throughput is $1/e$, and a sufficient and necessary condition to achieve this throughput is derived, which cannot always be satisfied.

In addition, for systems with TCP timeout backoff, due to the long duration of backoff periods, the system always operates at a range with too many idle slots, and hence has a throughput far below $1/e$. Whereas for systems without TCP timeout backoff, adjusting the MAC layer transmission probability can change the system load. In particular, when the system operates with too many collisions, such as systems with small propagation delay and large number of users, the throughput of $1/e$ can always be achieved by lowering the MAC layer transmission probability. However, in systems with large propagation delay and small number of users, the throughput of $1/e$ is not always achievable, because even when the transmission probability is set to its maximum value of 1, the system remains under-loaded. Moreover, since MAC layer retransmissions react faster to collisions and hence inject more traffic into the channel, MAC layer retransmissions are preferred for systems with light load. In contrast, when the system is heavily loaded, retransmissions at the transport layer (TCP) result in higher throughput.

Finally we consider the optimal scheduling problem with window service constraints. The objective is to minimize the average response time. In particular, we investigate in detail the effects of window constraints on packet service pattern, and develop an optimal and a more insightful suboptimal scheduling policy. The results show that both the job lengths and the window sizes are essential to the optimal policy. In addition, instead of changing priority of jobs at different times, in most cases the optimal policy gives full priority to one job.

# Bibliography


[1] The ns manual. *http://www.isi.edu/nsnam/ns/doc.*

[2] N. Abramson. the ALOHA system - another alternative for computer communications. In *Proceedings of Fall Joint Computer Conference, AFIPS Conference*, 1970.

[3] M. Allman, S. Dawkins, D. Glover, J. Griner, D. Tran, T. Herderson, J. Heidemann, J. Touch, H. Kruse, S. Ostermann, K. Scott, and J. Semke. Ongoing TCP research related to satellites. *Internet RFC 2760*, February 2000.

[4] M. Allman, C. Hayes, H. Kruse, and S. Ostermann. TCP performance over satellite links. In *Int. Conf. on Telecommunication Systems*, March 1997.

[5] Y. Bai, A.T. Ogielski, and G. Wu. Interactions of TCP and radio link ARQ protocol. In *Proceedings of IEEE Vehicular Technology Conference*, volume 3, page 1710, 1999.

[6] A. Bakre and B.R. Badrinath. I-TCP: Indirect TCP for modible hosts. In *15th International Conference on Distributed Computing Systems (ICDCS)*, May 1995.

[7] H. Balakrishnan, V.N. Padmanabhan, S.Seshan, and R.H. Katz. A comparison of mechanisms for improving TCP performance over wireless links. *ACM/IEEE Transactions on Networking*, December 1997.

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

[9] N. Bansal and M. Harchol-Balter. Analysis of SRPT scheduling: Investigating unfairness. *ACM SIGMETRICS Performance Evaluation Review*, 29, 2001.

[10] C. Barakat, E. Altman, and W. Dabbous. On TCP performance in a heterogeneous network: a surveys. *IEEE Communications Magazine*, 38:40, January 2000.

[11] P. Baran. On distributed communication networks. *IEEE Transactions on Communication Systems*, March 1964.

[12] D. Bertsekas and R. Gallagher. *Data Networks.* Prentice Hall, Upper Saddle River, NJ, 1992.

[13] R. Caceres and L. Iftode. Improving the performance of reliable transport protocols in mobile computing environments. *IEEE Journal on Selected Areas in Communications*, 13:850, 1995.

[14] A.F. Canton and T. Chahed. End-to-end reliability in UMTS: TCP over ARQ. In *Proceedings of IEEE GLOBECOM'01*, volume 6, page 3473, 2001.

[15] J.I. Capetanakis. The multiple access broadcast channel: protocol and capacity considerations. *IEEE Transactions on Information Theory*, 25, 1979.

[16] H.M. Chaskar, T.V. Lakshman, and U. Madhow. TCP over wireless with link level error control: analysis and design methodology. *ACM/IEEE Transactions on Networking*, 7(5), October 1999.

[17] C.F. Chiasserini and M. Meo. Modeling interactions between link layer and transport layer in wireless networks. In *12th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, volume 1, 2001.

[18] C.F. Chiasserini and M. Meo. A reconfigurable protocol setting to improve TCP over wireless. *IEEE Transactions on Vehicular Technology*, 51:1608, November 2002.

[19] A. Chockalingam, M. Zorzi, and V. Tralli. Wireless TCP performance with link layer FEC/ARQ. In *Proceedings of IEEE ICC'99*, volume 2, page 1212, 1999.

[20] A. DeSimone, M.C. Chuah, and O.C. Yue. Throughput performance of transport-layer protocols over wireless LANs. In *Proceedings of IEEE GLOBECOM'93*, December 1993.

[21] K. Fall and S. Floyd. Simulation-based comparisons of Tahoe, Reno, and SACK TCP. *Computer Communication Review*, 1996.

[22] R. Gallagher. *Discrete stochastic processes.* Kluwer Academic Publishers, Boston/Dordrecht/ London, 1995.

[23] P.E. Green. *Computer network architectures and protocols.* Plenum Press, New York and London, 1982.

[24] P.E. Green. Computer communications: milestones and prophecies. *IEEE Communications Magazine*, 22, May 1984.

[25] B. Hajek and T. Van Loon. Decentralized dynamic control of a multiaccess broadcast channel. *IEEE Transactions on Automatic Control*, 27, 1982.

[26] T.R. Henderson. *Networking over Next-Generation Satellite Systems.* PhD thesis, University of California Berkeley, 1999.

[27] T.R. Henderson and R.H. Katz. Transport protocols for internet-compatible satellite networks. *IEEE Journal on Selected Areas in Communications*, 17:326, February 1999.

[28] V. Jacobson and M. J. Karels. Congestion avoidance and control. In *Proceedings of Sigcomm*, 1988.

[29] E.J. Weldon JR. An improved selective-repeat ARQ strategy. *IEEE Transactions on Communications*, November 1993.

[30] P. Karn and C. Partridge. Improving round trip time estimates in reliable transport protocols. *Computer Communication Review*, August 1987.

[31] L. Kleinrock. Information flow in large communication networks. Technical report, RLE Quertly Progress Report, July 1961.

[32] L. Kleinrock. *1964 Communication Nets: Stochastic Message Flow and Delay.* McGraw-Hill, New York, 1964.

[33] L. Kleinrock and F.A. Tobagi. Packet switching in radio channels: part 1: CSMA modes and their throughput-delay characteristics. *IEEE Transactions on Communications*, 23, 1975.

[34] A.G. Konheim. A queuing analysis of two ARQ protocols. *IEEE Transactions on Communications*, July 1980.

[35] J.F. Kurose and K.W. Ross. *Computer Networking: A Top-Down Approach Featuring the Internet.* Addison Wesley, Boston, 2001.

[36] T.V. Lakshman and U. Madhow. The performance of TCP/IP for networks with high bandwidth-delay products and random loss. *IEEE Transactions on Networking*, June 1997.

[37] S. Lam and L. Kleinrock. Packet switching in multiaccess broadcast channel: dynamical control procedures. *IEEE Transactions on Communications*, 23, 1975.

[38] L.P. Larry and S.D. Bruce. *Computer Networks: a Systems Approach.* Morgan Kaufmann, San Francisco, 2000.

[39] F. Lefevre and G. Vivier. Optimizing UMTS link layer parameters for a TCP connection. In *Proceedings of IEEE Vehicular Technology Conference*, page 2318, 2001.

[40] C. Liu and E.H. Modiano. On the interaction of layered protocols: The case of window flow control and ARQ. *Conference on Information Science and System*, March 2002.

[41] C. Liu and E.H. Modiano. On the performance of TCP congestion control over satellite links with ARQ. *Computer Networks, to appear*, 2004.

[42] S. Muthukrishnan, Rajmohan Rajaraman, Anthony Shaheen, and Johannes E. Gehrke. Online scheduling to minimize average stretch. In *Proc. 40th Annual IEEE Symp. on Foundations of Computer Science*, page 433, 1999.

[43] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP throughput: a simple model and its empirical validation. In *Proceedings of ACM SIGCOMM '98 conf. on Applications, technologies, architectures, and protocols for computer communication*, volume 28, October 1998.

[44] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP Reno performance: a simple model and its empirical validation. *IEEE/ACM Transactions on Networking*, 8:133, April 2000.

[45] C. Partridge and T.J. Shepard. TCP/IP performance over satellite links. *IEEE Network*, 11:44, 1997.

[46] L. Schrage. A proof of the optimality of the shortest remaining processing time discipline. *Operation Research*, 16(3):687, 1968.

[47] L. Schrage and L. W. Miller. The queue M/G/1 with the shortest remaining processing time discipline. *Operation Research*, 14(5):670, 1966.

[48] D. R. Smith. A new proof of the optimality of the shortest remaining processing time discipline. *Operation Research*, 26(1):197, 1978.

[49] J.S. Stadler. A link layer protocol for efficient transmission of TCP/IP via satellite. In *Proceedings of IEEE MILCOM'97*, page 723, 1997.

[50] J.S. Stadler and J. Gelman. Performance enhancement for TCP/IP on a satellite channel. In *Proceedings of IEEE MILCOM'98*, page 270, 1998.

[51] W. Stallings. *Data and Computer Communications.* Macmillan, New York, 1994.

[52] W.R. Stevens. *TCP/IP Illustrated*, volume 1. Addison Wesley, New York, 1994.

[53] F.L.H.M. Stumpers. The history, development, and future of telecommunications in Europe. *IEEE Communications Magazine*, 22, May 1984.

[54] D. Towsley and J.K. Wolf. On the statistical analysis of queue lengths and waiting times for statistical multiplexers with ARQ retransmission schemes. *IEEE Transactions on Communications*, March 1982.

[55] B.S. Tsybakov. Survey of USSR contributions to random multiple-access communications. *IEEE Transactions on Information Theory*, 31, 1985.

[56] A. Wierman and M. Harchol-Balter. Classifying scheduling policies with respect to unfairness in an M/GI/1. *ACM SIGMETRICS Performance Evaluation Review*, 31, 2003.

[57] A.S. William. *Understanding Data Communications and Networks.* ITP, London, 2000.

[58] J.W.K. Wong and V.C.M. Leung. Improving end-to-end performance of TCP using link-layer retransmissions over mobile internetworks. In *Proceedings of IEEE ICC'99*, page 324, 1999.

[59] R. Yavatkar and N. Bhagawat. Improving end-to-end performance of TCP over mobile internetworks. In *Proceedings of Workshop on Mobile Computing Systems and Applications*, December 1994.

[60] M. Yoshimoto, Y. Takahashi T. Takine, and T. Hasegawa. Waiting time and queue length distributions for go-back-N and selective-repeat ARQ protocols. *IEEE Transactions on Communications*, November 1993.

[61] L. Zhang. Why TCP timers don't work well. In *Proceedings of ACM SIGCOMM conf. on Communications architecture and protocols*, 1986.

[62] H. Zimmermann. OSI reference model - the ISO model of architecture for open systems interconnection. *IEEE Transactions on Communications*, 28:425–432, April 1980.

[63] M. Zorzi, R.R. Rao, and L.B. Milstein. ARQ error control for fading mobile radio channels. *IEEE Transactions on Vehicular Technology*, May 1997.