IMPROVING END-TO-END PERFORMANCE OF TRANSMISSION CONTROL PROTOCOL (TCP) USING LINK LAYER RETRANSMISSIONS OVER MOBILE INTERNETWORKS

by

JACKSON W. K. WONG

Bachelor of Engineering in Electronics Engineering

The Hong Kong University of Science and Technology, 1995

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF APPLIED SCIENCE

in

THE FACULTY OF GRADUATE STUDIES

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

We accept this thesis as conforming to the required standard

THE UNIVERSITY OF BRITISH COLUMBIA

APRIL 1998

© Jackson W. K. Wong, 1998

In presenting this thesis in partial fulfilment of the requirements for an advanced degree at the University of British Columbia, I agree that the Library shall make it freely available for reference and study. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by the head of my department or by his or her representatives. It is understood that copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Department of ELECTRICAL AND COMPUTER ENGINEERING

The University of British Columbia
Vancouver, Canada

Date 17 APR 1998

Abstract

TCP does not perform well in networks with high packet error rate like those with wireless links because TCP assumes network congestion to be the major cause for packet losses. Wireless losses make TCP unnecessarily initiate congestion control mechanism which results in poor performance in the form of low throughput and high interactive delay. The link layer scheme, which employs data link protocols in the base station and mobile host to retransmit lost packets over wireless link, may be employed to hide wireless losses from TCP, but the problem of competing retransmissions between TCP and link layer may occur, causing unnecessary duplications and significant degradation in TCP performance.

This thesis investigates, through computer simulations, the end-to-end effects of link layer retransmissions over a low data-rate wireless link on TCP Reno. The results show that, by using the more effective selective-reject ARQ in the link layer, the problem of competitive retransmissions between TCP and link layer is much less serious than previously reported. It is also found that a non-sequencing link layer in combination with fragmentations of datagrams at the base station and mobile host can be employed without significantly degrading TCP performance, thus avoiding re-sequencing buffers and complex logic for handling out-of-sequence packets that would otherwise be needed for a sequencing link layer protocol. The link layer modifications for best-effort retransmissions, with a suitable division of the wireless-loss recovery function between TCP and link layer, are proposed to reduce the possible adverse effects of link layer resets and increased round trip time estimates from link layer recovery on TCP. The effects of link layer reliability, controlled by the maximum number of link layer retransmissions or the maximum link layer recovery time, on TCP throughput and round trip time estimations are studied.

Table of Contents

List of Tables

List of Figures

Acknowledgment

I would like to express my sincere gratitude to my research supervisor, Prof. Victor C.M. Leung for his valuable suggestions throughout the course of my research. I am grateful to my family and friends for the constant encouragement and support. I would also like to thank Motorola for its financial support and MIL 3 Inc. for providing the access to the OPNET simulator. Finally, thanks to my fellow students, William Mok, Lewis Chen and Eva Leung for their helpful comments on my thesis.

Chapter 1 Introduction

In this introductory chapter, a brief history and the architecture of the current Internet communication (TCP/IP) is reviewed in section 1.1. The objectives and motivations of this research are mentioned in section 1.2. The approach that is employed to achieve the objectives is explained in section 1.3. The findings and contributions of this work are outlined in section 1.4. Finally, a roadmap of the thesis is given in section 1.5.

1.1 Background

The Internet employing the Transmission Control Protocol / Internet Protocol (TCP/IP), developed in the late 1960s as a research project conducted by the United States Department of Defense, is a network architecture designed to tie together an extremely large number of computer networks in both military and academic organizations scattered across the continent. Since it was not possible to provide direct connectivity between any pair of hosts in such a large network, the best way to efficiently accomplish this scale of connectivity was to model the underlying structure as a *packet-switched network.*

In the *packet-switched* network model, each node on the network only has direct connections to its neighboring nodes and can only communicate *directly* with these nodes. Communications to any other host must take place *indirectly* through several intermediate hosts, called routers or gateways. The information which is intended to pass from the source host to the destination host is carried by a number of packets (formally called *datagrams*). Each datagram has attached with it the address of the destination host. The address uniquely identifies the host to which the datagram is destined but says nothing about the route which must be taken to get there. The route is instead determined by the intermediate routers through which the datagram passes. Each router

has a limited knowledge of its *local* environment and is able to intelligently choose one of its immediate neighbors as the best next step to which the datagram should be forwarded. This forwarding of the datagram continues, with local routing decision being made at each intermediate router, until eventually the datagram is passed to the destination host.

In order to support such a packet-switched network, TCP/IP is organized as a set of functional layers [1][2] (see Figure 1.1). Within each layer, a well defined set of network services are provided, service which in turn utilizes the services provided by its lower layers. This structure is similar to the fundamental concept in the design of the OSI model [1].

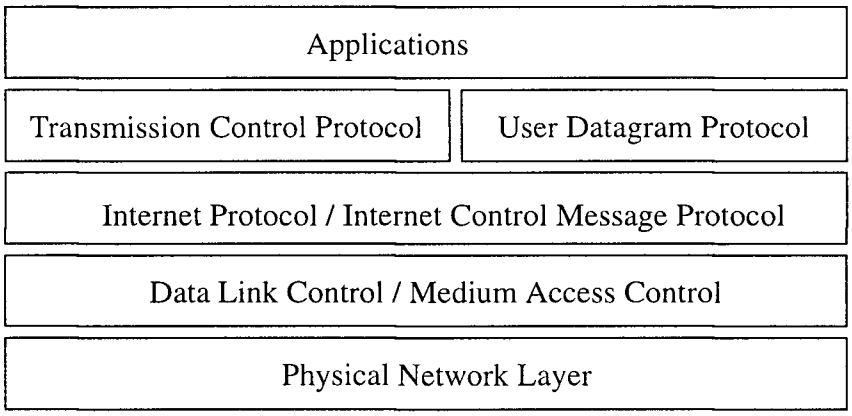

Figure 1.1 Communication architectures of TCP/IP

The Physical layer covers the physical interface between neighboring nodes and the rules by which bits are passed from one node to another. The Data Link Control (DLC) layer attempts to make the physical link more reliable and provides means to activate, maintain and deactivate a communication connection. The Medium Access Control (MAC) layer controls the access to the physical transmission medium in order to efficiently utilize its capacity.

The Internet Protocol (IP) [3] governs how individual datagrams can be delivered over the network. Each datagram contains the *IP address* of the destination host. When a host receives a

datagram which is addressed to *another* host, its IP performs the routing decision based on the IP address in the datagram. The Internet Control Message Protocol (ICMP) [4] handles the error reporting associated with the IP.

The User Datagram Protocol (UDP) [5] provides the user with direct access to the services of the lower IP layer. Datagrams transmitted by UDP are guaranteed neither delivery nor correctness. Unlike UDP, the Transmission Control Protocol (TCP) [6] provides a reliable transport service, that guarantees the correct in-sequence delivery of data, to the applications. Residing at the highest level of the TCP/IP structure are a set of network applications or utility programs. Several well known TCP/IP applications are TELNET, a terminal emulation program used to establish remote login sessions, FTP, a file transfer protocol which supports the high speed transfer of both text and binary data between hosts, and HTTP, an information-retrieval protocol that supports data, voice, image and movie transfer in the World Wide Web (WWW).

1.2 Objectives and Motivations

Nowadays, the *Internet* provides another form of medium for global communication and information sharing. Recent tremendous growth on the number of mobile phone users indicates that the integration of our current technology on wireless communication and computer networking will play an important role in the future Internet. With the arrival of portable computers as powerful as some desktop workstations in terms of computing power, memory, display and disk storage, people expect to carry their laptop computers with them whenever they move from place to place and yet maintain network access to the global Internet through *wireless links*.

Unfortunately, the initial design of the current Internet protocol suite, TCP/IP, did not take the mobile users and wireless links into consideration. The existing versions of TCP such as RFC

793 [6] and TCP Reno [7][8][9] assume *network congestion* to be the major cause for packet losses. In response to *any* packet loss, TCP initiates the congestion control mechanism (to be explained in section 2.1.4) to avoid swamping the network with additional packets after congestion occurs. However, in the presence of high packet error rate in the wireless links, this mechanism slows down the traffic transmitted over the TCP connection and causes TCP to suffer a significant performance degradation in form of low throughput and very high interactive delay [10]. This behavior arises because wireless errors are incorrectly interpreted by the TCP sender as network congestion losses.

The current TCP/IP networks need to be modified to accommodate the wireless environment. In order to *seamlessly* integrate mobile devices communicating on wireless links with the Internet, it is important not to make any modification to the existing TCP implementation in the *fixed* networks. It is even better to use the same TCP implementation in the *wireless* networks. Since the only parts of the mobile Internet we have full administrative control over are the wireless networks, we hope to make modifications solely to the base stations and mobile hosts of the wireless network.

A **link layer** scheme, which employs *data link protocols in the base stations and mobile hosts* to retransmit lost packets over the wireless link, attempts to hide wireless losses from TCP by making a lossy wireless link appear as a more reliable link. Therefore most of the losses seen by TCP are caused by network congestion. The intuition behind this scheme is that since the problem is local in the wireless link, it should be solved locally and TCP needs not be aware of the existence of a wireless link. The advantage of this scheme is that it fits naturally into the layered structure of TCP/IP (see Figure 1.1) without requiring changes to the existing TCP implementa-

tion in the fixed or mobile hosts.

The data link layer, however, needs to be tuned to the characteristics of the wireless link to provide good throughput performance and efficient link utilization. Optimization of several key parameters in the data link protocols may be needed so as to match the data rate and delay characteristics of the wireless link.

The disadvantage of the link layer scheme is that since the end-to-end TCP connection passes through the wireless link, TCP may not be fully shielded from wireless losses. The problem of competing retransmissions between TCP and data link layer *may* occur when the TCP retransmit timer (to be explained in section 2.1.2) times out before the data link layer recovers a wireless loss. Therefore both TCP and data link layer retransmits the lost packet, producing unnecessary duplications in TCP. In [11], analysis of a TCP-like transport protocol with a *stop-and-wait* [1] link layer protocol by simulations shows that link layer retransmissions improve TCP throughput only when the packet error rate (PER) of the wireless link is higher than 10%. At smaller PERs, almost all the packets retransmitted by the data link layer are also retransmitted by TCP. These unnecessary duplications due to competing retransmissions contribute to the waste of valuable capacity in the wireless link and significant degradation of end-to-end TCP throughput.

The objectives of this thesis are:

- to analyze the end-to-end effects of link layer retransmissions over a wireless link on TCP Reno (the current de facto standard for TCP) and discover possible adverse effects of link layer retransmissions on TCP,
- to investigate the possibility of competing retransmissions between TCP and data link layer using the more effective selective-reject ARQ [1] in the link layer,

- to evaluate several retransmission schemes in data link layer and optimize the key parameters used in these data link protocols,
- to explore the end-to-end effects of re-sequencing in data link layer on TCP Reno, and
- to examine the TCP performance improvement of using best effort retransmissions for link layer recovery of wireless losses.

1.3 Approach

The approach to achieve the above research objectives is outlined as follows:

1. Formulate a network model of the mobile Internet using the OPNET [12] network simulation tool.
   - Model the Reno version of TCP.
   - Model the route decision making, fragmentation and reassembly of datagrams in the intermediate IP routers.
   - Model several commonly implemented retransmission schemes in the data link layer.
   - Model the error characteristics of the wireless link and the medium access delay at the MAC layer of the wireless network.
2. Identify the key parameters of the data link protocols that affect the performance of TCP and obtain their optimized values under different packet error rates in the wireless link using OPNET simulations.
3. Analyze the end-to-end effects of link layer retransmissions on the round trip time estimations (to be explained in section 2.1.3), throughput and delay performance of TCP and investigate the possibility of competing retransmissions between TCP and data link layer.
4. Modify the OPNET models of the data link protocols to include both sequencing and non-

sequencing frame delivery and evaluate the TCP performance between these two approaches.

5. Modify the OPNET models of the data link protocols so that they implement best-effort retransmissions and evaluate the outcomes of the best-effort approach on TCP.

1.4 Contributions of the Thesis

This thesis investigates the competing retransmission problem for TCP Reno over a slow speed wireless link typical in a mobile data network and determines how link layer retransmissions can be fine-tuned to minimize competing retransmissions in TCP.

Using the OPNET network simulation tool, the end-to-end effects of different retransmission schemes and link layer protocol parameters on the throughput and delay performance of TCP are evaluated. Instead of the stop-and-wait scheme used in [11], high performance link layer retransmission schemes based on go-back-N and selective-reject are considered. Simulation results show that link layer recovery effectively increases the round trip time (RTT) estimates and the retransmission time-out (RTO) value of TCP, which reduces the likelihood of TCP retransmission time-outs and hence the possibility of competitive retransmissions with the link layer.

The investigation uncovered another problem that a link layer scheme can cause in some existing protocols like CDPD MDLP [13]. If link layer retransmissions of each data frame are limited to a small number, bad channel condition can cause frequent link layer resets, resulting in packet losses and TCP performance degradation. However, while increasing the maximum number of link layer frame retransmissions for each packet minimizes the occurrences of link layer resets, it results in large RTO estimates in TCP which could severely delay recovery of congestion losses by time-out retransmissions. We propose a best-effort approach in which the

link layer does most of the recovery for wireless losses and TCP does the rest, without resetting the data link connection. Simulation results show that with a suitable choice of the maximum number of link layer retransmissions for each packet, the increase in RTO can be controlled and the adverse effect of link layer recovery can be minimized.

The effects of link layer re-sequencing on TCP performance are studied by simulations. The results suggest that non-sequencing data link protocols in combination with fragmentations of datagrams at the base station and mobile host can be employed for wireless loss recovery without significantly degrading TCP performance. Instead, re-sequencing buffers and complex logic for handling out-of-sequence packets that would otherwise be needed for sequencing data link protocols can be avoided, thus simplifying the protocol implementation.

1.5 Outline of the Thesis

In chapter 2, an overview of the TCP Reno is given, and the problem of the TCP Reno in the wireless environment and several related works to solve the problem are reviewed. Chapter 3 describes the simulation model for the mobile Internet. Chapter 4 presents the simulation results and the performance analysis of TCP using link layer retransmissions over a wireless link. The comparison of TCP performance with and without in-sequence delivery at the data link layer is also discussed there. In chapter 5, the best-effort modifications for link layer protocols and the analysis of the relations between link layer reliability and TCP behavior are given. Chapter 6 summarizes all the findings in this research and suggests future investigations to further improve TCP performance in the wireless environment.

Chapter 2 The Mobile Internet

In this chapter, an overview of the mobile Internet is provided. Some key concepts of TCP Reno (the current de facto standard of TCP in the Internet) are briefly discussed in section 2.1. The problems of the mobile Internet are pointed out in section 2.2. The link layer scheme and several related works that are proposed to solve the problem are outlined in section 2.3.

2.1 Overview of TCP Reno

The functions of TCP Reno [7][8][9] are described in the following five subsections. TCP Reno uses a selective-reject [1] scheme to provide in-sequence delivery to the higher layer. For simplicity, TCP Reno is referred to as *TCP* throughout this and all the following chapters.

2.1.1 Segmentation in TCP

A TCP application sends a message by passing stream of data bytes to TCP. TCP internally buffers the byte stream and breaks the buffered byte stream into *segments* before transmissions, regardless of how the original data stream was received from the sending application. This is different from UDP, where each service call by the application generates a UDP datagram with size equal to the total size of the application data and the UDP header.

Each TCP segment contains the source and destination *port numbers* of the TCP applications. These two values, along with the source and destination IP addresses in the IP header, uniquely identify the **source application** in the **source host** and the **destination application** in the **destination host**. Every TCP segment contains a *sequence number*, a 32-bit unsigned number that wraps back around to 0 after reaching 2^{32} - 1, which identifies the byte number in the stream of data that the first byte of the data in this segment represents.

In order to minimize the cost of the TCP/IP headers, each time TCP sends the biggest possible data segment from the buffered byte stream. The maximum size of a TCP segment is referred to as the **Maximum Segment Size (MSS)**, which can be announced by the TCP in both ends of the connection during connection establishment. After all, TCP should send with an MSS value only up to the Maximum Transfer Unit (MTU) of the outgoing interface minus the total size of the TCP/IP headers so as to avoid fragmentation.

2.1.2 Error Recovery and Flow Control in TCP

TCP uses a **positive** acknowledgment mechanism with time-out retransmission to enable error recovery. When TCP receives a data segment from the other end of the connection, it *schedules* to send an acknowledgment (ACK) back so as to inform the sender the successful arrival of the segment. An ACK contains an acknowledgment number which is the next sequence number that the sender of the ACK expects to receive. TCP ACK is *cumulative* in the sense that sending an ACK number N means receipts of all data bytes up to sequence number N-1.

An ACK is not sent immediately, but normally delayed a maximum time period called **Maximum ACK Delay**. Under the idea of *delayed ACK*, if there is a data segment going in the same direction as the ACK during that period of time, the ACK will be piggybacked into the header of the outgoing data segment. Sending a piggybacked ACK costs nothing more because the 32-bit ACK field is always part of a TCP header. If the time period passes before an outgoing data segment is sent, a dataless ACK will be sent.

Loss recovery in TCP is performed by time-out retransmissions at the sender. When TCP sends a data segment, it stores the byte stream of the data segment into a retransmission buffer and starts a *retransmit timer*. Arrival of an ACK to this data segment clears the byte stream in the

buffer and stops the retransmit timer. If the retransmit timer times out before the ACK is received, the TCP sender will retransmit the data segment.

TCP uses a sliding window mechanism to implement flow control. The sliding window mechanism allows the receiver to restrict the transmission of the sender until it has sufficient buffers to receive more data. The receiver advertises the available capacity of its receive buffer to the sender in each outgoing ACK. The size of the receive buffer, called **Receiver Window Size**, is a TCP parameter which is set when TCP is installed in a network. Under the sliding window mechanism, the total number of bytes of unacknowledged segments in the retransmission buffer must be smaller than the window size advertised by the remote receiver. Therefore a data segment cannot be sent if the total size of the data segment and all unacknowledged segments is larger than the remote window size. When this occurs, we say that "the remote window is closed". Further data segment can be sent only if an ACK is received that clears some unacknowledged segments and "opens the window".

2.1.3 Round Trip Estimation in TCP

The retransmission time-out (RTO) value set at the retransmit timer is crucial to the effectiveness of the retransmission mechanism. If the RTO is too small, packets may be retransmitted too frequently and therefore unnecessarily. On the other hand, if the RTO is too large, loss recovery by time-out retransmission may be too slow. TCP uses frequent measurements of the round trip time (RTT) to dynamically adjust the RTO. To estimate the RTT, TCP measures the time it takes for a segment to be acknowledged. Then it uses the sequence of RTT samples, s_1, s_2, s_3, ..., to calculate the "smoothed" RTT (SRTT) and the mean deviation of the SRTT (Dev). For each new sample, s_i , the SRTT and Dev are updated as follows:

$$\begin{aligned} SRTT_{i+1} &= SRTT_i + g \cdot (s_i - SRTT_i) \\ &= (1-g) \cdot SRTT_i + g \cdot s_i \end{aligned} \tag{2.1}$$

$$\begin{aligned} Dev_{i+1} &= Dev_i + h \cdot (|s_i - SRTT_i| - Dev_i) \\ &= (1-h) \cdot Dev_i + h \cdot (|s_i - SRTT_i|) \end{aligned} \tag{2.2}$$

where $SRTT_i$ is the current estimate of RTT and $SRTT_{i+1}$ is the newly calculated estimate. Similarly, Dev_i is the current estimate of the mean deviation of SRTT and Dev_{i+1} is the newly calculated estimate. The RTT gain g and RTT deviation gain h are constants between 0 and 1 that control how fast SRTT and SRTT deviation, respectively, adapt to changes in network traffic. g is recommended to be 1/8 and h is recommended to be 1/4 in [14].

The retransmission time-out (*RTO*) is assigned to be the sum of the smoothed round trip time (*SRTT*) and n times the mean deviation of the smoothed round trip time (*Dev*). That is,

$$RTO_{i+1} = SRTT_{i+1} + n \times Dev_{i+1} \tag{2.3}$$

where the RTT Deviation Coefficient, n, is set to 4 in [14]. Compared with the older versions of TCP, TCP Reno has a coarse retransmission time-out granularity that is typically multiples of 500ms, which means the RTO values are rounded off to the nearest multiples of 500ms.

When an ACK is received for a segment that has been retransmitted, there is no indication on which transmission is being acknowledged. This problem is known as retransmission ambiguity. To solve the problem, TCP Reno employs Karn's algorithm [15], which requires that round trip estimates be updated only for the ACKs of those segments that have been transmitted only once.

2.1.4 Congestion Control in TCP

When network congestion occurs, delay increases and intermediate routers begin to queue up packets and eventually start to drop packets. When the network delay exceeds a certain limit, TCP retransmit timer expires which causes more segments to be transmitted and the network to be more congested. This avalanche-like situation causes the network to breakdown and reduces the throughput to almost zero. To avoid congestion collapse, TCP must *reduce transmission rate* when network congestion, which is signalled by a time-out at the TCP sender, is detected.

TCP uses an exponential retransmit timer back-off scheme [14] for time-out retransmission. Each time a *consecutive* time-out takes place, the RTO is doubled. As a result, the rate at which retransmission packet is injected in the network is reduced to half after each successive time-out. Because of the Karn's algorithm, TCP does not update the RTO when a retransmission takes place and therefore the RTO grows exponentially in each successive time-out. Two other techniques are also employed to deal with congestion: the slow start and congestion avoidance.

In addition to the remote receive window, TCP maintains a state variable, called *congestion window*, to control congestion. At any time the size of the transmit window is the minimum of the remote receive window and the congestion window. When the network is not congested, the congestion window equals the remote receive window. However, when the retransmit timer expires, the **slow start mechanism** [14] sets the congestion window to one MSS (which reduces the transmit window to one MSS) in order to prevent the sender from adding more packets to the congested network. When congestion ends and ACKs start to flow back to the sender, the slow-start mechanism increases the congestion window by one MSS each time an ACK is received.

Contrary to what the term slow start suggests, under ideal conditions the start is *not* very

slow. Initially, TCP sends one segment and waits. When the ACK arrives, it increases the congestion window to 2 MSS, allowing two segments to be sent before waiting. When the two ACKs arrive, each increases the congestion window by 1 MSS, allowing TCP to send 4 segments. Thus the congestion window experiences *exponential growth* after slow start. To avoid increasing the transmit window too quickly and causing additional congestion, the **congestion avoidance mechanism** [14] increases the congestion window by 1 MSS only if all segments in the transmit window have been acknowledged, which gives a *linear growth* in the congestion window.

TCP Reno [7] uses the **combined slow start with congestion avoidance algorithm** [14] for congestion control. In addition to the congestion window (*cwnd*), the sender keeps another state variable called the threshold size (*ssthresh*). The algorithm uses these two variables to switch between slow start and congestion avoidance. When the sender times out, one half of its current transmit window is recorded in *ssthresh* and *cwnd* is set to 1 MSS. When new data is acknowledged, the sender increases *cwnd* by 1 MSS if *cwnd* is less than or equal to *ssthresh*. If *cwnd* is larger than *ssthresh*, *cwnd* is increased by MSS x MSS / *cwnd*. Thus **slow start** opens the congestion window quickly but once the congestion window reaches one half of its original size, **congestion avoidance** takes over and slows down the rate of incrementing the congestion window.

2.1.5 Fast Retransmit and Fast Recovery in TCP

Losses caused by network congestion can be detected in two situations:

1. a time-out at the TCP sender, or
2. receipts of several consecutive ACKs with the same ACK number, called *duplicate ACKs*.

In the first situation, all the transmitted segments are probably lost and slow starts takes place to re-start the transmission (according to section 2.1.4). However, in the second situation,

some of the transmitted segments arrives at the receiver out-of-sequence. Realizing that TCP generates an *immediate* ACK when an out-of-order segment is received, this duplicate ACK should not be delayed. The purpose of this duplicate ACK is to let the sender know that a segment was received out-of-order and what next sequence number is expected.

Segments following different routes in the Internet experience variable delay, which causes segments to be received out-of-order. Since TCP needs to provide in-sequence delivery to its application, a TCP receiver has buffers to store out-of-sequence segments for reordering. As we do not know whether a duplicate ACK is caused by a lost segment or just a reordering of segments, we wait for a small number of duplicate ACKs to be received. Normally, if there is just a reordering of segments, only one or two duplicate ACKs are received before the reordering segment has arrived, which generates a new ACK. Therefore we can reasonably assume that some segment is lost when *three* or more consecutive duplicate ACKs are received.

Besides the combined slow start with congestion avoidance algorithm, TCP Reno implementation has a Fast Retransmit and Fast Recovery algorithm [7]. With Fast Retransmit, after receiving three duplicate ACKs, the TCP sender infers that a segment is lost and re-transmits the missing segment without waiting for its retransmit timer to expire. Fast Retransmit enables TCP to detect and retransmit a lost segment more quickly than time-out retransmission. The Fast Recovery algorithm operates by assuming each duplicate ACK received represents a single segment arriving at the receiver. Therefore each duplicate ACK increments the congestion window by 1 MSS, thereby allowing the sender to send more packets and prevents the communication pipe from going empty after Fast Retransmit. The Fast Retransmit and Fast Recovery algorithm operates as follows [7][8]:

1. When the first and second duplicate ACKs arrive, the sender waits.
2. When the third duplicate ACK is received, one half of the current value of congestion window (*cwnd*) is saved in *ssthresh*. Since receiving three duplicate ACKs means three segments have arrived at receiver, *cwnd* is set to *ssthresh*+3MSS. The segment with sequence number equal to the acknowledgment number of the duplicate ACK is re-transmitted.
3. Each time another duplicate ACK arrives, *cwnd* is increased by 1 MSS. New packet can be sent if allowed by the new value of *cwnd*.
4. When an ACK that acknowledges the missing segment is received, *cwnd* is set back to *ssthresh*. This "recovery ACK" should be the acknowledgment of the retransmission segment and possibly some subsequent out-of-sequence segments. Eventually the flow is slowed down to one half the rate when the segment is detected lost.

2.2 TCP Reno under wireless environment

TCP was originally designed for an environment where the packet error rate (PER) is low. However, in the wireless environment, noise bursts, multipath fading and interference result in much higher PER due to relatively frequent transmission errors. Worst still, TCP interprets wireless errors as signs of network congestion. In response to congestion as mentioned before, TCP initiates the combined slow start with congestion avoidance algorithm to slow down the traffic transmitted over the connection because the algorithm drops the congestion window and restricts the rate at which the congestion window grows to previous levels. The exponential retransmit timer back-off scheme for time-out retransmission can further delay the error recovery. As a result, TCP suffers from significant throughput degradation and unacceptable interactive delay over a wireless environment.

Nevertheless, these algorithms are found to be useful in the current Internet. The major problem is that TCP cannot distinguish between wireless losses and congestion losses. Recently, some works have been proposed to lessen or eliminate the effects of *non-congestion-related* losses on TCP performance over networks with wireless links.

2.3 The Link Layer scheme and some Related works

A **link layer** scheme may be employed to hide wireless losses from TCP by using forward error correction (FEC) [1] and/or retransmission of lost packets over the wireless link using an automatic repeat request (ARQ) protocol [1]. Examples of commercial system that use the link layer approach are CDPD, which uses a LAPD-derived link layer protocol called MDLP [13], and the CDMA wireless system [16]. The main advantage of employing a link layer protocol for recovery of wireless losses is that its implementation is confined to the data link layer at the base stations and mobile hosts, so that it fits naturally into the layered structure of TCP/IP without requiring changes to TCP.

Several other schemes have been suggested to reduce or eliminate the adverse effect of wireless losses on TCP. However, they all have their limitations.

The **end-to-end** scheme attempts to make TCP distinguish between congestion losses and wireless losses using an *Explicit Loss Notification (ELN)* mechanism [17]. The end-to-end protocol belongs to a wireless-aware transport protocol which requires TCP to know whether its intended receiver is connected to a wireless link or not. Thus this scheme needs to modify existing TCP implementation in both fixed and mobile hosts.

The **split-connection** scheme, on the other hand, completely hides the wireless link from

TCP in fixed hosts by terminating their (fixed) TCP connections at the base stations of the wireless network. Such scheme uses a separate transport protocol for the connection between base station and mobile host. An example is Indirect TCP (I-TCP) [18] which uses a separate (wireless) TCP connection over the wireless link, in concatenation with the corresponding fixed TCP connection. The disadvantage of this approach is that it violates the end-to-end semantics of TCP ACKs, since an ACK for a segment sent over the first connection can reach the sender even before the segment forwarded over the second connection actually reaches the destination.

The **snoop protocol** [19] introduces a snoop agent at the base station, which monitors every segment and ACK that passes through a TCP connection in both directions. The agent maintains a cache of the TCP segment sent across the wireless link that has not yet been acknowledged by the TCP receiver and retransmits the segment from the cache if the segment is detected lost. This protocol suppresses duplicate ACKs for lost TCP segments and retransmits them locally over the wireless link, thereby avoiding unnecessary fast retransmit and fast recovery invocations by the TCP sender. However it does not improve the performance of any transport protocol other than TCP and it does not work particularly well for data transfers from mobile hosts to fixed hosts.

Considering the above alternatives, it is apparent that the link layer approach is the best choice. The main problem about the link layer approach is the possibility of competing retransmissions between TCP and link layer [11]. This can happen when the TCP retransmit timer times out before the link layer recovers a wireless loss, so that both TCP and link layer retransmits the lost data, an unnecessary duplication. We investigate this behavior in chapter 4 and show that, by using the more effective selective-reject ARQ, the problem of competitive retransmissions between TCP and link layer is much less serious than previously reported in [11].

Chapter 3 The Model of Mobile Internetworks

This chapter describes a model of the mobile Internet. Section 3.1 outlines the situation of a typical mobile cellular network. The simulation models of each functional layer in the communicating entities are provided in section 3.2. Section 3.3 discusses why fragmentation of packets over the wireless link should be employed. A brief overview of the OPNET simulation tool is given in section 3.4.

3.1 Model of the Mobile Cellular Networks

The mobile Internet can be considered as a mixture of fixed networks, the global Internet and wireless networks. A logical view of a mobile cellular network and the manner in which it interfaces with the global Internet are shown in Figure 3.1.

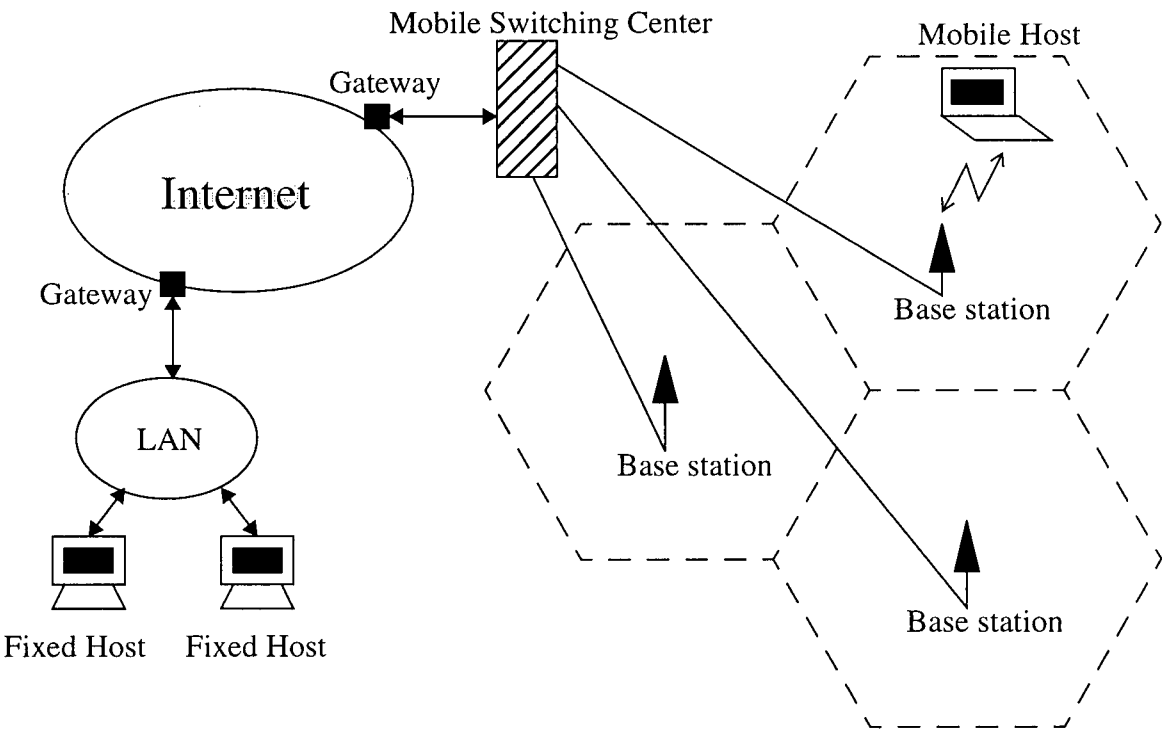

Figure 3.1 Model of a mobile Internetwork

The gateways and the Internet perform routing functions for data packets transferred between a fixed network and a wireless network. The fixed network may be a local area network (LAN) connecting to a certain number of fixed hosts. The wireless network may be a mobile cellular network, which is divided into location areas called cells. Each cell has a base station which manages the wireless interfaces with mobile hosts. All base stations are connected to a switching office, called the mobile switching center (MSC). At any moment a MSC keeps a record of the cells where the mobile hosts are located. When the MSC receives a packet destinated for a mobile host that it currently controls, it uses the location information and forwards the packet to the appropriate base station through which the mobile host can be reached.

Since a mobile host moves freely in the area covered by its wireless network, it can get out of reach of its current base station by entering any of its neighboring cells. This situation is called *hand-off*. Location update on the mobile host may be involved in the MSC and the base stations during or after a hand-off. The details of some hand-off schemes can be found in [20][21].

Because of multipath fading and co-channel interference, the bit error rate (BER) of the wireless link can be as large as 10^{-1}, causing a high packet error rate (PER). Besides, packets may be lost and communications on the wireless link may pause during hand-offs. The hand-off duration depends on the speed of the mobile host, the size of the overlapping region between the two neighboring cells where hand-off takes place and the type of hand-off scheme. As a result, the wireless environment may introduce a high degree of unreliability in the transmission of packets between a fixed host and a mobile host.

3.2 Network Models

In order to evaluate the performance of TCP with link layer recovery between the base station and mobile host, a TCP connection between a mobile host and a fixed host incorporating a wireless data link connection is considered. Figure 3.2 shows the communication architecture of the mobile Internet in Figure 3.1 and the protocol layers that a TCP segment goes through in order to be delivered between the fixed host and the mobile host. The following subsections present the network models of the protocol layers and elements in Figure 3.2. The order of description follows an upward direction from the lowest to the highest layer.

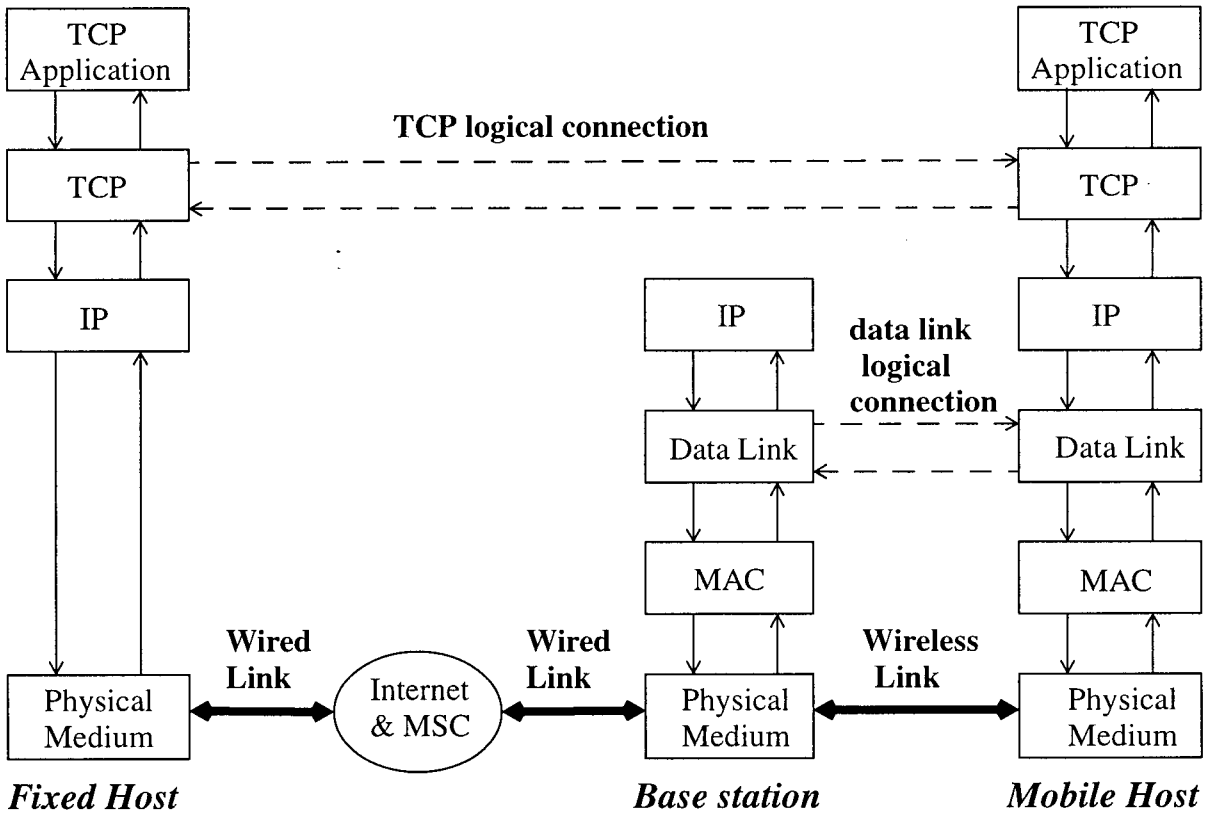

Figure 3.2 Communication architecture of a TCP connection between a fixed host and a mobile host

3.2.1 The Physical Medium

The physical medium is the lowest layer in Figure 3.2 which provides the physical interfaces between the fixed host, the base station and the mobile host. The data rates of the communication links are shown in Table 3.1. All communication links are assumed to be *full-duplex*. Previous work on improving TCP performance in wireless networks was mostly focused on wireless LANs [11][19], which have data rates of 1-2Mbps. With increasing use of laptop computers and portable digital assistants, there is a growing demand for wireless Internet access over a wide area, using mobile data networks which typically employ slow speed wireless links with data rates from 2.4Kbps to 19.2Kbps. A wireless data rate of 19.2Kbps is considered in this work.

Wired link between fixed host and Internet	10 Mbps
Wired link between Internet and base station	10 Mbps
Wireless link between base station and mobile host	19.2 Kbps

Table 3.1 Data rates of the communication links

The propagation delay of the wired link is negligible when compared with the end-to-end delay introduced by the Internet. The propagation delay in the wireless link represents the time it takes for radio wave to travel between the base station and the mobile host, which is much smaller than the transmission delay of the slow wireless link. Thus, the propagation delays of all communication links are ignored in the model.

Since the BER on the wired link is relatively smaller than the BER on the wireless link (up to 10^{-1}), the two wired links are assumed to be error-free. The bit error characteristics of wireless links have been observed to be bursty due to fading effects which are described by the fading rate and average fade duration [22]. With a high wireless data rate, the packet duration is generally shorter than the average fade duration, causing several successive packet losses in a deep fade.

However, a low wireless data rate is considered in this work, with which the packet duration can be an order of magnitude longer than the fade duration and thus a deep fade occurs mostly within a single packet. For simplicity, an uniform error model with a wide range of PERs from 0 to 60%, the same as the previous work in [11], is used.

3.2.2 Medium Access Control (MAC) layer

A mobile host may use a separate frequency channel or share a certain frequency band with other mobile hosts. If a frequency band is shared by a number of mobile hosts, some medium access entity should exist to control the use of wireless channel. Examples of medium access control (MAC) schemes are Aloha, FDMA, TDMA and CDMA. Figure 3.3 shows a wireless channel shared between a number of mobile hosts and a single base station.

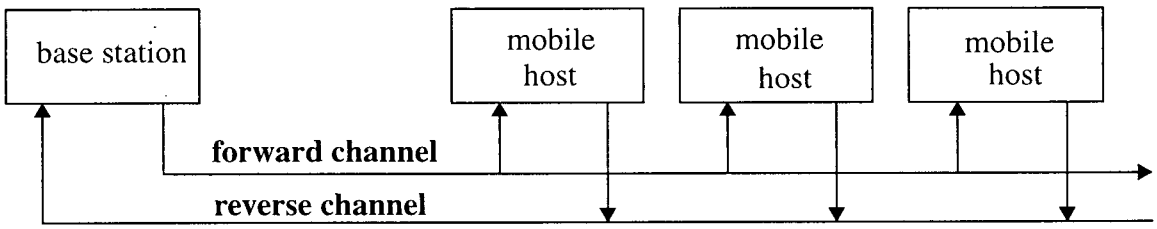

Figure 3.3 Model of a shared wireless channel

The wireless channel composes of a *forward channel* from base station to mobile hosts and a *reverse channel* from mobile hosts to base station. The forward channel is a contentionless broadcast channel carrying transmission from the base station only. Information is received by all mobile hosts on the forward channel simultaneously. The reverse channel is shared among the mobile hosts. Arbitration of access and resolution of contention can be controlled by either the base station or the mobile hosts, depending on the type of MAC technology.

The MAC layer attempts to reliably deliver packets in sequence without loss, duplication

or corruption. However, due to the inherent noisy nature of the wireless medium or possibly due to channel congestion, some frames may not be delivered error-free. Frames in error are discarded and any further recovery action is the responsibility of the higher layers (e.g. data link layer).

Data link layer and MAC layer exchange primitives to provide signalling between each other [23]. When the data link layer has a frame to send, it uses a Request primitive to request service from the MAC layer. When the MAC layer has successfully delivered a frame, it uses a Confirm primitive to inform the data link layer that the transmission has been completed. When the MAC layer receives a frame, it uses an Indication primitive to inform the data link layer the arrival of a data link layer frame. The data link layer can use the Request and Confirm primitives to enable flow control of frames to the MAC layer. For instance, if the data link layer is allowed to forward only one frame to the MAC layer at any time, the data link layer should wait for a Confirm primitive before sending a second Request primitive.

Frame arriving at the MAC layer may wait for a certain period of time before the MAC can gain access to the wireless channel. The MAC delay relies on the number of mobile hosts sharing the wireless channel, the traffic load of the mobile hosts and the type of MAC technology used. In [24], the distributions of the MAC delay in Slotted ALOHA and Carrier Sense Multiple Access (CSMA) are derived by formulating Markovian models with finite populations of users, each with a single packet buffer. The packet delay of Carrier Sense Multiple Access with Collision Detect (CSMA/CD) is found to be exponentially-distributed by using a continuous-time Markov chain model [25]. Exact models of several MAC protocols is not employed in this work. Instead, a more general approach is to use the MAC delay to model the MAC layer. The MAC delay is modeled as an exponential distribution. For simplicity, packet losses due to excess channel access attempts are not taken into considerations.

3.2.3 Data Link layer

The data link layer provides a logical data link connection across the wireless link between a base station and a mobile host and utilizes the services of the MAC layer to provide transparent transfer of link layer frames between two communicating link layer entities. This work considers both a *connection-oriented* data link, in which packet sequencing is preserved at the receiving network layer and transmission service is reliable, and a *connectionless* data link, in which packets are immediately delivered to the receiving network layer when correctly received over the data link, without regard to packet sequencing, and reliability is not guaranteed.

While the channel stream of the wireless link is *point-to-multipoint* (see Figure 3.3), the data link connection is *point-to-point* (see Figure 3.4). Broadcast information transfer to mobile hosts by data link layer is not considered. Each mobile host requires a point-to-point data link connection to communicate with the base station. *Direct* communication between two mobile hosts using a data link is not possible.

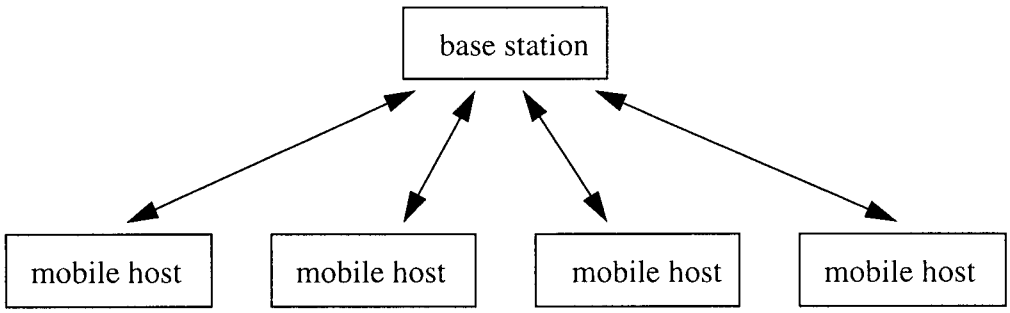

Figure 3.4 Point-to-point data link connections between base station and mobile hosts

Link layer frames delivered over the wireless link may be lost or damaged. A *lost* frame is a frame that fails to reach the other end of the connection, probably due to a noise burst which damages the frame to the extent that the receiver is not aware of the frame being transmitted. A

damaged frame is a recognizable frame that arrives the receiver but some of its bits are in error. Here we assume that the MAC layer does not pass any damaged frame to the data link layer. Since the data link layer does not receive any damaged frame, it detects a loss by receiving a subsequent frame which is out-of-sequence.

Three types of services provided by the data link layer are considered: unreliable, reliable and best-effort. An unreliable link layer does not include any error recovery or flow control mechanism. With reliable delivery, link layer frames are transmitted with sequence numbers, error recovery procedures based on retransmissions are involved and flow control by acknowledgment may be included. With best effort delivery, error recovery is performed only to a limited extent. In case of a lost frame that cannot be recovered by the data link layer after a few number of tries, data link layer gives up delivery of the frame.

In this work, error recovery of data link layer is solely performed by retransmission. FEC, which decreases the error probability of frames by adding some parity bits to correct the bit errors as many as possible [1], is usually considered a physical layer function.

To implement flow control, the data link layer uses a sliding window mechanism similar to TCP (section 2.1.2), which allows the sender to transmit a certain number of frames before receiving an acknowledgment. The number of outstanding frames should be smaller than or equal to the window size. Data link layer assigns each packet received from the higher layer a sequence number. While the sequence number in TCP identities a byte in the stream of data packet (section 2.1.1), the sequence number in the data link layer identifies the whole packet. The sequence number of each link-layer frame is an n-bit unsigned number that wraps back around to 0 after reaching $2^n - 1$.

To provide error recovery, the receiver returns positive acknowledgments for successfully received frames and negative acknowledgments for lost frames. In addition, the sender retransmits a frame that has not been acknowledged after a predetermined amount of time, called the retransmission time-out. Loss recovery of the data link layer can be initiated either by the sender, receiver or both sides. In the first case, the sender is the only side responsible for recovery, possibly by retransmission after time-out. In the second case, no time-out retransmission is employed at the sender. Instead the receiver initiates loss recovery by requesting retransmissions via negative acknowledgments. The third case represents a combination of both methods.

There are many data link protocols developed. Different protocols require different amount of storage buffer, computing power and logic complexity. Simple protocols like stop-and-wait, go-back-N and selective-reject ARQ can be found in [1]. An example of a more complex protocol that is designed for error-prone wireless links is the AIRMAIL scheme [26], which uses a combination of FEC and ARQ. An asymmetric protocol that uses two different data link protocols at the base station and mobile host is designed to reduce the processing overheads in the mobile host [27]. As link layer protocols employing stop-and-wait ARQ is poor in efficiency, a link layer protocol employing go-back-N and two link layer protocols employing selective-reject are considered: the LAPB, the CDPD MDLP and the RLP.

3.2.3.1 LAPB

The Link Access Protocol - Balanced (LAPB) [28], a HDLC subset used in X.25 packet switching networks, uses a go-back-N retransmission scheme. LAPB uses 8-bit or 128-bit sequence numbers. Acknowledgments can be piggybacked onto information frames (I frames). The receiver starts a T2 timer to schedule the return of a positive acknowledgment whenever an I

frame is received. An explicit positive acknowledgment (RR) frame is sent if the T2 timer expires before an I frame is sent.

When an out-of-sequence frame is received, LAPB sends a negative acknowledgment frame (REJ) with a sequence number indicating the next frame expected to be received. Once the receiver sends a REJ, it enters into a reject recovery state, in which no more REJ will be sent and all subsequent out-of-sequence frames received are discarded, thus requiring no re-sequencing buffer. When the sender receives the REJ, it reverts to retransmitting all unacknowledged frames, starting at the one specified by the REJ (i.e. go-back-N). The receiver exits the reject recovery state only if the expected in-sequence frame is received.

The sender uses a T1 timer to trigger a time-out retransmission and restarts the T1 timer whenever a frame is transmitted or retransmitted. When the T1 timer expires before an acknowledgment for any outstanding frame is received, the sender starts retransmission beginning at the first unacknowledged frame. When a REJ or a retransmitted I frame is lost, the T1 timer at the sender will eventually time out. As a result, loss recovery in LAPB can be initiated by both the sender and receiver. When a REJ fails to recover a loss, the sender will do the rest of the recovery by time-out retransmission using the T1 timer. Table 3.2 shows a list of LAPB parameters that may affect link layer recovery performance.

T1 Timer Time-out Value (i.e. Retransmit Timer Time-out)
T2 Timer Time-out Value (i.e. Maximum Acknowledgment Delay)
Transmit Window Size

Table 3.2 LAPB parameters

3.2.3.2 CDPD MDLP

The CDPD Mobile Data Link Protocol (MDLP) [13], modified from HDLC, implements a selective-reject ARQ. It uses 8-bit or 128-bit sequence numbers. Acknowledgments can be piggybacked onto information frames (I frames). The receiver starts the T205 timer to schedule the return of a positive acknowledgment whenever a frame is received. An explicit positive acknowledgment (RR) frame is sent if the T205 timer expires before an I frame is sent.

When an out-of-sequence frame is received, MDLP stores the frame in a re-sequencing buffer and transmits a negative acknowledgment frame (SREJ) to initiate an exception condition recovery for *each* newly detected missing I frame. Any number of SREJ exception conditions for a given direction of information transfer may be established at a time and thus the receiver can request more than one retransmission at a time. Normally a SREJ is not retransmitted after loss. If multiple SREJ conditions are outstanding, it may be possible to detect the need to retransmit a SREJ using the algorithm mentioned in [13]. However the algorithm cannot detect the need to re-send the last SREJ. This re-sending of SREJ is optional in the MDLP specification.

Similar to LAPB, the MDLP sender uses a T200 timer to trigger time-out retransmission and loss recovery in MDLP can be initiated by both the sender and receiver. When a SREJ fails to recover a loss, the sender will do the rest of the recovery by time-out retransmission using the T200 timer. Table 3.3 shows the MDLP parameters that may affect link recovery performance.

T200 Timer Time-out Value (i.e. Retransmit Timer Time-out)
T205 Timer Time-out Value (i.e. Maximum Acknowledgment Delay)
Transmit Window Size

Table 3.3 MDLP parameters

3.2.3.3 RLP

The Radio Link Protocol (RLP), modified from [27], is a selective-reject scheme in which recovery is initiated only by the receiver. Acknowledgments are not piggybacked onto information frames. Instead the receiver periodically sends status feedback frames to cumulatively acknowledge all packets received in-sequence or request retransmissions of specific packets indicated by a bitmap when losses are detected. The use of a bitmap in the retransmission request reduces the overheads of individual negative acknowledgments. The periodic status feedback minimizes the adverse effect of errors in the return channel [27]. In particular, if the channel is poor and loses all information frames, status feedback frame can still be sent because sending of status feedback is triggered by periodic time-out rather than by the event of receiving an information frame. Moreover, if a status message is lost, there is always another one coming next. Note that the period of the status timer can be adjusted so as not to waste too much wireless capacity for status feedback frames while at the same time minimizing the effect of a noisy channel.

There is *no time-out retransmission* at the RLP sender. If the *last* frame sent by the sender is lost, the retransmission request sent by the receiver will never cover the loss since no subsequent out-of-sequence frame will be received to detect the loss. The RLP is designed such that whenever the sender receives a status feedback, it checks all unacknowledged frames if they should be retransmitted. The sender does not retransmit all unacknowledged frames but rather retransmits a frame only if this frame was sent a long time before, the time which is defined as the *minimum time between transmissions,* so as to ensure that this frame is really lost rather than delayed when the status feedback is sent by the receiver. The value of the minimum time between transmissions can be adjusted so as to avoid too many redundant retransmissions. Note that the minimum time between transmissions differs from the normal retransmission time-out (in LAPB

and MDLP) because there is no time-out retransmission in RLP, but both of them can control the time between successive retransmissions.

Table 3.4 shows the RLP parameters that may affect link recovery performance.

Period of Receiver Status Feedback Timer
Minimum Time Between Transmissions
Transmit Window Size

Table 3.4 RLP parameters

3.2.4 IP layer, the Internet and the Mobile Switching Center (MSC)

Below TCP lies the IP layer [3]. TCP requests service from IP by passing the TCP segment, and the IP address of the remote host to IP. IP then encapsulates the TCP segments with the IP addresses of the local host and the remote host to form an *IP datagram*. Whenever IP receives a datagram, it uses the encapsulated IP address to make routing decision. Since the IP of the fixed host and mobile host receives packets both from TCP and the lower layer, the IP layer uses the encapsulated IP address to determine whether an incoming packet is destinated to the local host or a remote host.

All IP modules introduce delay during routing decision making and datagram transmission. The IP in each communicating entity is modeled as a FIFO queue, providing service to both segments coming from higher and lower layers. Table 3.5 shows the service rates of IP at each node. The values are stated here solely for modeling purpose. The exact values do not contribute much effect on the TCP throughput and end-to-end delay since the IP routing delay is considered to be small when compared with the transmission delay over the wireless link.

Fixed Host	5000 packets per second
Base station	1000 packets per second
Mobile host	1000 packets per second

Table 3.5 IP service rates

The Internet can be considered as a series of routers which make the path to deliver an IP datagram from the gateway at the fixed network to the gateway at the Mobile Switching Center in Figure 3.1. The Internet delay is modeled by a sum of a constant of 0.02s and an Erlang distributed random variable with mean 0.01s [10]. The characteristics of the congestion losses in the Internet are modeled by the Fritchman binary error model [29], which captures the burstiness of congestion losses using a two-state Markov model. This basic error model contains two states: Congested and Clear (see Figure 3.5). In the Congested state, datagrams routing through the Internet are lost due to congestion. In the Clear state, datagrams routing through the Internet are successfully delivered.

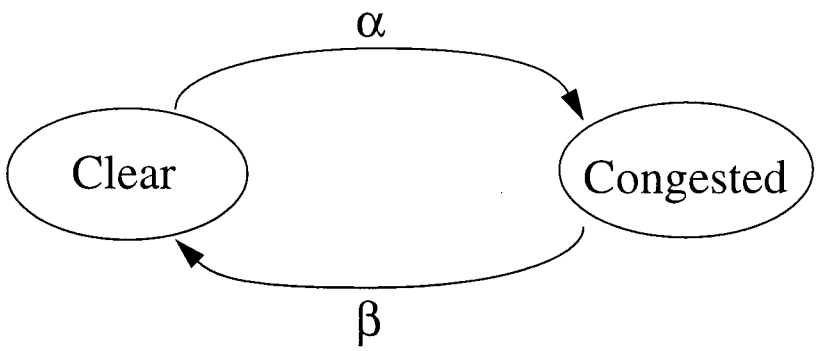

Figure 3.5 Fritchman binary error model for congestion losses

The average duration of staying in each state is expressed in term of transition rate. The transition rates from the Clear state to the Congested state and from the Congested state to the Clear state are denoted as α and β respectively.

The average duration of congestion is $\frac{1}{\beta}$ and the average time between congestions is $\frac{1}{\alpha}$.

The average number of congestion periods per second is $\frac{1}{\frac{1}{\alpha}+\frac{1}{\beta}} = \frac{\alpha \cdot \beta}{\alpha+\beta}$.

The equilibrium probabilities of staying in Clear state (p_L) and in the Congested state (p_C) are given respectively by

$$p_L = \frac{\beta}{\alpha+\beta}$$
$$p_C = \frac{\alpha}{\alpha+\beta} \qquad \text{Eqn (3.1)}$$

The datagram loss rate in the Internet is therefore given by

$$p_C = \frac{\alpha}{\alpha+\beta} \qquad \text{Eqn (3.2)}$$

The Mobile Switching Center (MSC) determines the base station through which an IP datagram can reach the mobile host. For simplicity, the modeling of the MSC is not included.

3.2.5 TCP Reno and TCP applications

On top of TCP stands the TCP applications, such as FTP, Telnet, Rlogin. When an application want to use TCP, it requests a full-duplex TCP connection from its local TCP. Once a TCP connection is established, the TCP application can then forward bytes of data to TCP. The library module of TCP in the OPNET [12] simulator is modified to model TCP Reno [8][9], using the flow control, error control and congestion control mechanisms of TCP Reno mentioned in Chapter 2.

Table 3.6 shows the parameters used for TCP Reno. Although TCP Reno uses a 16384-byte receiver window, a smaller window size is chosen because a large window gives a large round trip time over a low data-rate wireless link, increasing the end-to-end TCP delay and delaying TCP error recovery. Most TCP implementations use a 200 ms Maximum ACK Delay and a Maximum Segment Size (MSS) of 536 bytes [8]. A value of 512 bytes for MSS is used so as to make it an integral fraction of the window size. The values of RTT and the RTO parameters are as suggested in [14].

Receiver Window Size at fixed host	4096 bytes
Receiver Window Size at mobile host	4096 bytes
Maximum ACK Delay	0.2 second
Maximum Segment Size	512 bytes
RTT Gain	0.125
RTT Deviation Gain	0.25
RTT Deviation Coefficient	4.0
Initial SRTT	1.0 second
Initial SRTT Deviation	1.0 second
Minimum RTO	2.0 seconds
Maximum RTO	240.0 seconds

Table 3.6 TCP Reno parameters

TCP throughput and TCP end-to-end delay are the measures used for performance evaluation and are defined in chapter 4. In order to measure the maximum sustainable throughput, TCP is provided with application data whenever it needs so as to fill up the transmission pipe as much as possible. In each simulation, at least 40 Mbits of bulk data are sent from the application layer in the source host to the application layer of the remote host so as to make sure that the measured throughput reaches equilibrium.

3.3 Fragmentation of IP datagrams at the base station and mobile host

Given a 512-byte MSS in TCP (see Table 3.6), with the addition of a 20-byte TCP header and 20-byte IP header [6][3], the maximum size of an IP datagram is 552 bytes or 4416 bits. The PER on the wireless link generally increases with the length of a packet and may be unacceptably high for large IP datagrams. However using a small MSS increases the relative overhead of the TCP/IP headers. As a compromise, large IP datagrams are fragmented over the wireless link. Since IP provides a fragmentation and reassembly mechanism [3] [30], fragmentation of IP datagrams is assumed to be performed at the IP layers of the base station and mobile host. The Maximum Transfer Unit (MTU) of IP at the basestation and mobile host is assumed to be 1024 bits, which is close to 130 octets as suggested in [13].

Fragmentation at IP is achieved by breaking the information part of an IP datagram into pieces [3] and encapsulating these pieces with the IP addresses of the original IP datagram and several fields that distinguish between fragments. Reassembly of IP fragments takes place when the IP fragments reach the destination [3][30]. For data transfer from the fixed host to the mobile host, fragmentation takes place at the base station and reassembly takes place at the mobile host. For data transfer from the mobile host to the fixed host, fragmentation takes place at the mobile host and reassembly takes place at the fixed host.

With an MSS of 512 bytes in TCP, the maximum size of a TCP segment is 532 bytes (i.e. 512 + 20) or 4256 bits. With an MTU of 1024 bits in IP, the maximum size of the information part of an IP datagram is 864 bits (i.e. 1024 - 20 x 8). Therefore, the number of IP fragments for a maximum-size TCP segment is 5 (i.e. 4256 / 864).

3.4 OPNET Simulation Models

The simulation models of the mobile Internet are constructed using the OPtimized Network Engineering Tools (OPNET) [12]. OPNET is a window-based simulation package with graphical user interfaces. Modeling in OPNET is divided by hierarchy into a set of three modeling domains:

- *Network Domain*: concerned with the specification of a network system in terms of nodes (communicating entities) and communication links between them.
- *Node Domain*: concerned with the specification of node capability in terms of applications, processing, queueing and communication interfaces.
- *Process Domain*: concerned with the specification of behavior for the processes that operate within the nodes of the system. Decision making processes and algorithms of protocol in each functional layer can be specified.

The network, node and process models of the mobile Internet model are provided in Appendix B.

Chapter 4 Performance Analysis of TCP with Link Layer Retransmissions in Wireless Environment

In this chapter the end-to-end effects of link layer retransmissions over a wireless link with a data rate of 19.2Kbps on TCP are discussed. The behavior of TCP without link layer recovery is mentioned for comparisons. Performance evaluation on TCP throughput and delay using different link layer retransmission schemes and data link protocol parameters is also given.

Section 4.1 analyzes the behavior of TCP over an error-prone wireless link, focusing on the retransmission mechanism of TCP and the dynamics of TCP congestion window. Examples are used to illustrate how time-out and fast recovery are caused by frequent wireless losses.

Section 4.2 considers the behavior of TCP with link layer retransmissions over the wireless link. The throughput and end-to-end delay performance of TCP without link layer recovery are compared with TCP using RLP (see section 3.2.3.3) for link layer recovery. The effects of link layer retransmissions on the round trip time estimations in TCP are analyzed. Possible problem of competing retransmissions between TCP and data link layer is examined.

In section 4.3, the TCP performance using data link protocols with in-sequence delivery are compared with TCP using non-sequencing data link protocols. The trade-off to choose between these two approaches and the effects of IP datagram fragmentations over the wireless link are discussed.

In section 4.4, receiver-initiated and sender-initiated retransmission schemes are analyzed. Simulations results of go-back-N and selective-reject link layer protocols using LAPB, CDPD MDLP and RLP are shown. The parameters of these protocols that affect TCP performance are

optimized by simulations.

The following definitions are used for performance evaluation in this chapter and the next:

- *Packet error rate (PER)* is defined as the average loss probability of a data link layer frame delivered over the wireless link.
- *Average transmission delay of data link layer* is defined as the difference between the time when a data link layer frame is first transmitted by the source data link layer and the time when the frame is successfully received by the destination data link layer.
- *Average end-to-end delay of TCP* is defined as the difference between the time when a TCP segment is first transmitted by the source TCP host and the time when the segment is successfully received by the destination TCP host.
- *TCP throughput* is defined as the number of bits of application data transferred by TCP divided by the total time taken.

4.1 Behavior of TCP in the wireless environment

Figure 4.1 demonstrates the retransmission mechanism of TCP and how wireless losses can cause TCP to retransmit after a time-out (as explained in section 2.1.2) and to invoke fast retransmit and fast recovery mechanism (as explained in section 2.1.5). Simulation results at a wireless PER of 1% show that TCP suffers a much larger recovery delay during time-out retransmission than during fast retransmit and fast recovery.

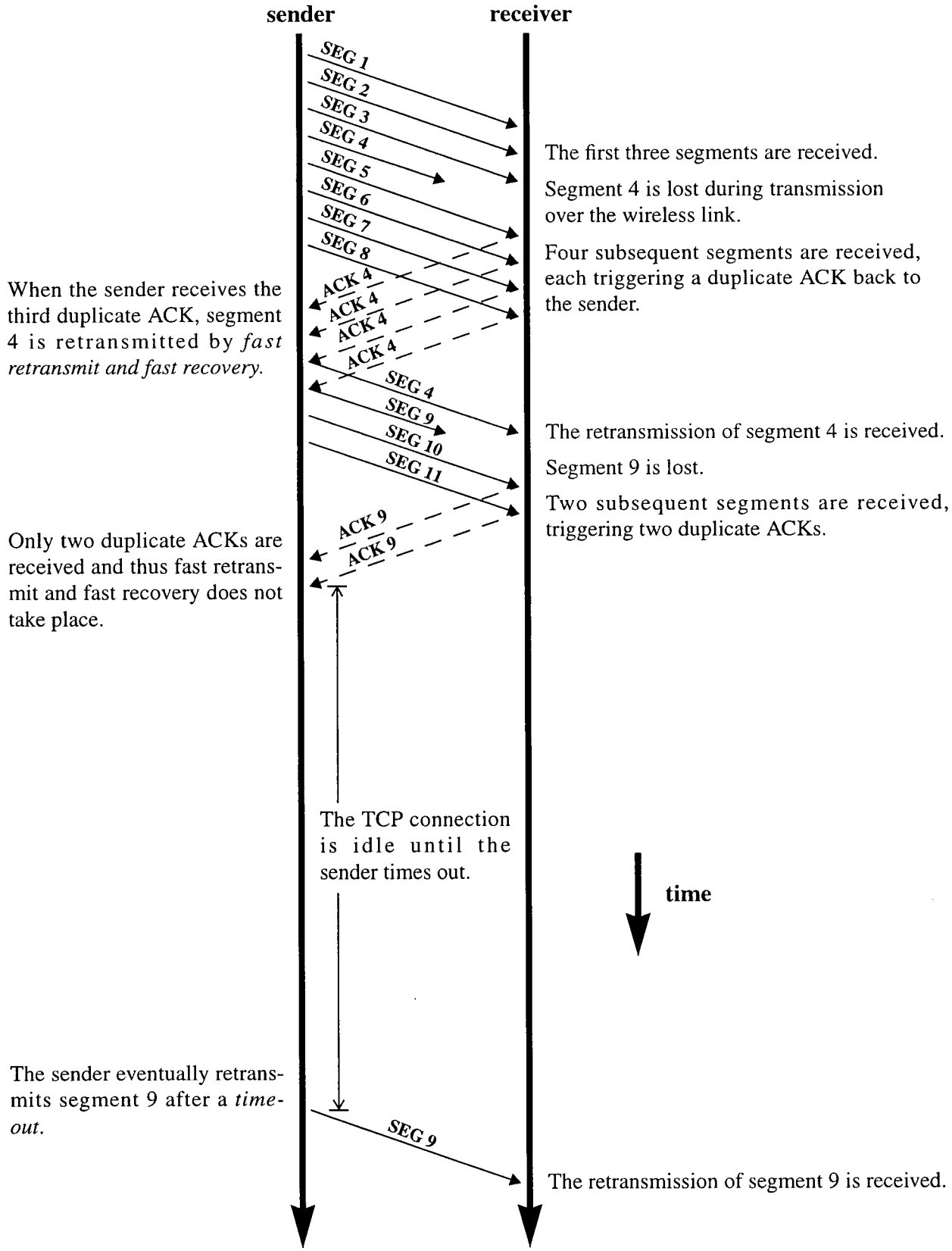

Figure 4.1 Illustration of fast retransmit / fast recovery and time-out retransmission in TCP

Figure 4.2 shows the dynamics of the TCP congestion window (as explained in section 2.1.4) of the illustration in Figure 4.1. The third fast retransmit / fast recovery and the time-out at 256.6s in Figure 4.2 corresponds to the fast retransmit / fast recovery and the time-out retransmission in Figure 4.1 respectively. In Figure 4.2, after three executions of fast retransmit / fast recovery, the congestion window is reduced to 1828 bytes at 252.1s, allowing only 3 (i.e. [1828/512]) 512-byte TCP segments to be transmitted. As indicated in Figure 4.1, segment 9 is lost. Receipts of two out-of-sequence segments, segments 10 and 11, can only generate two duplicate ACKs. As a result, the sender does not fast-retransmit segment 9, which has to be recovered by a time-out retransmission at 256.6s. Note that the congestion window is dropped to one MSS, i.e. 512 bytes, after the time-out. In this particular example, the time-out is caused by a loss within a small congestion window after several invocations of fast retransmit and fast recovery. During the time between 252.1s and 256.6s, the sender is idle and does not transmit any segment, which shows that TCP suffers long delay and large throughput degradation when time-out occurs.

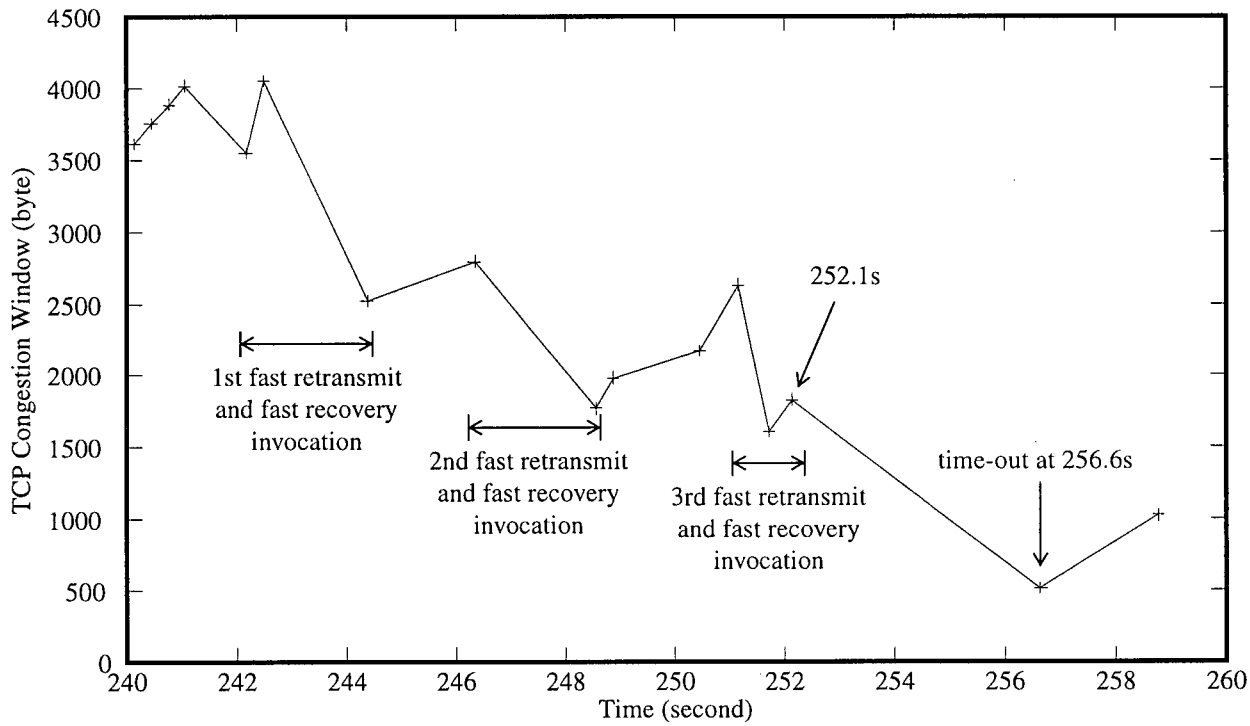

Figure 4.2 Congestion window at TCP sender when fast retransmit / fast recovery and time-out occur

Figure 4.3 illustrates how time-out occurs when two consecutive TCP segments are lost. The conclusion is that, when two or more segments are lost in the transmit window, even if the first segment is recovered by fast retransmit, the remaining ones need to be recovered by time-out retransmissions.

There are many other circumstances that can cause a time-out. For instance, when a fast-retransmit segment is lost, the TCP sender always times out because the TCP sender performs fast retransmit for a segment once only. Moreover, the loss of a time-out retransmission segment definitely produces another time-out. The TCP sender sometimes times out if the *ACK* to a retransmission segment is lost. Consider after receiving a retransmission segment, no more segment is received because the sender transmit window is closed and thus no more ACK can be generated. If the ACK of the retransmission is lost, the sender will misunderstand that the receiver has lost the retransmission segment and eventually retransmit the segment again after a time-out.

Figure 4.4 demonstrates the situation with two consecutive time-outs. A lost segment is fast-retransmitted at 210.1s but fails to reach the receiver. At this moment, the RTO estimate is 4s and thus the sender times out 4s later (at 214.1s). Again the time-out retransmission segment is lost and the next time-out occurs at 8s later (at 222.1s) because the exponential back-off scheme (as explained in 2.1.4) doubles the RTO after each consecutive time-out. With these two consecutive time-outs, the receiver is idle from 211.1s to 222.1s.

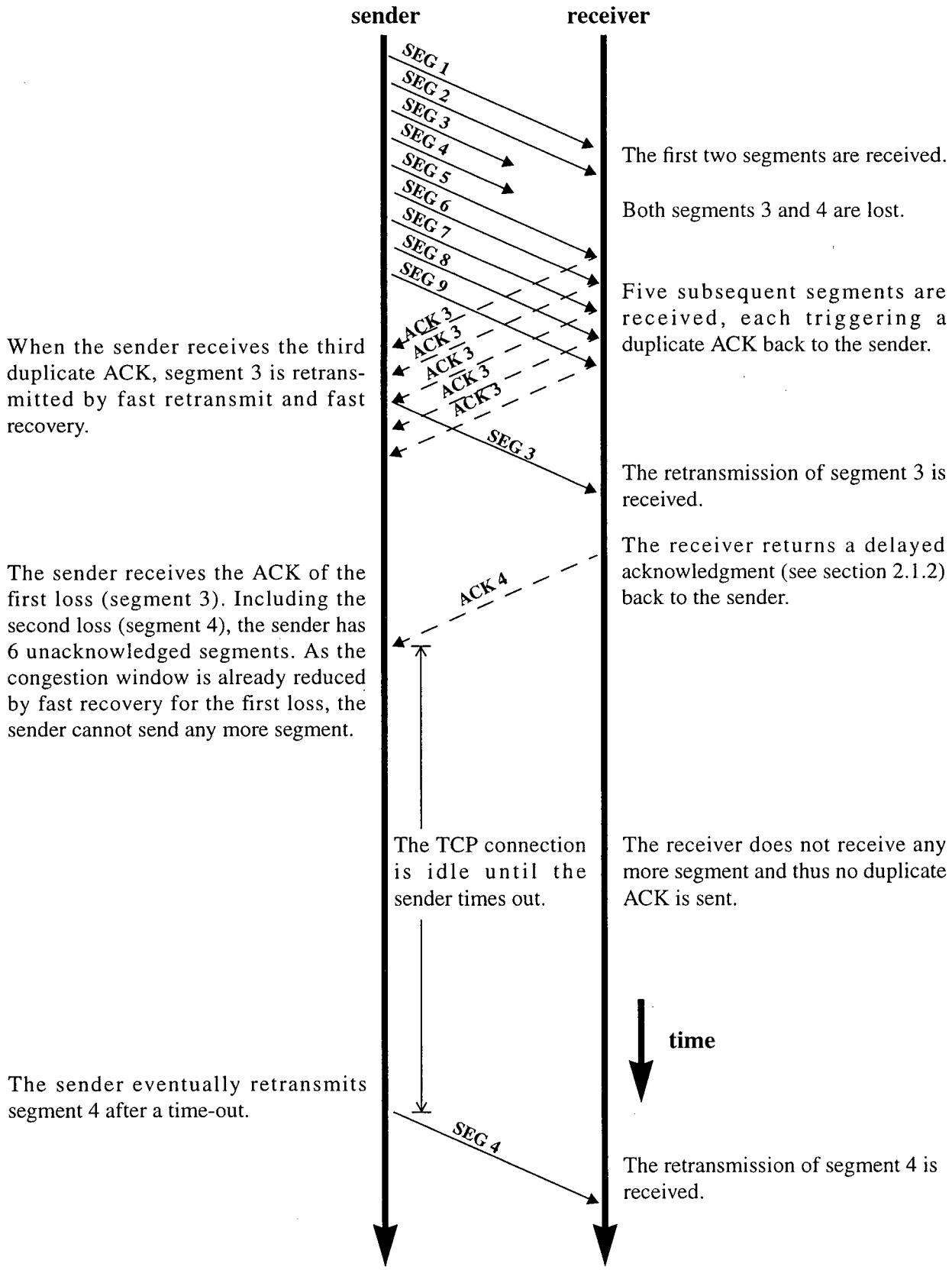

Figure 4.3 Illustration of two consecutive segment losses in TCP

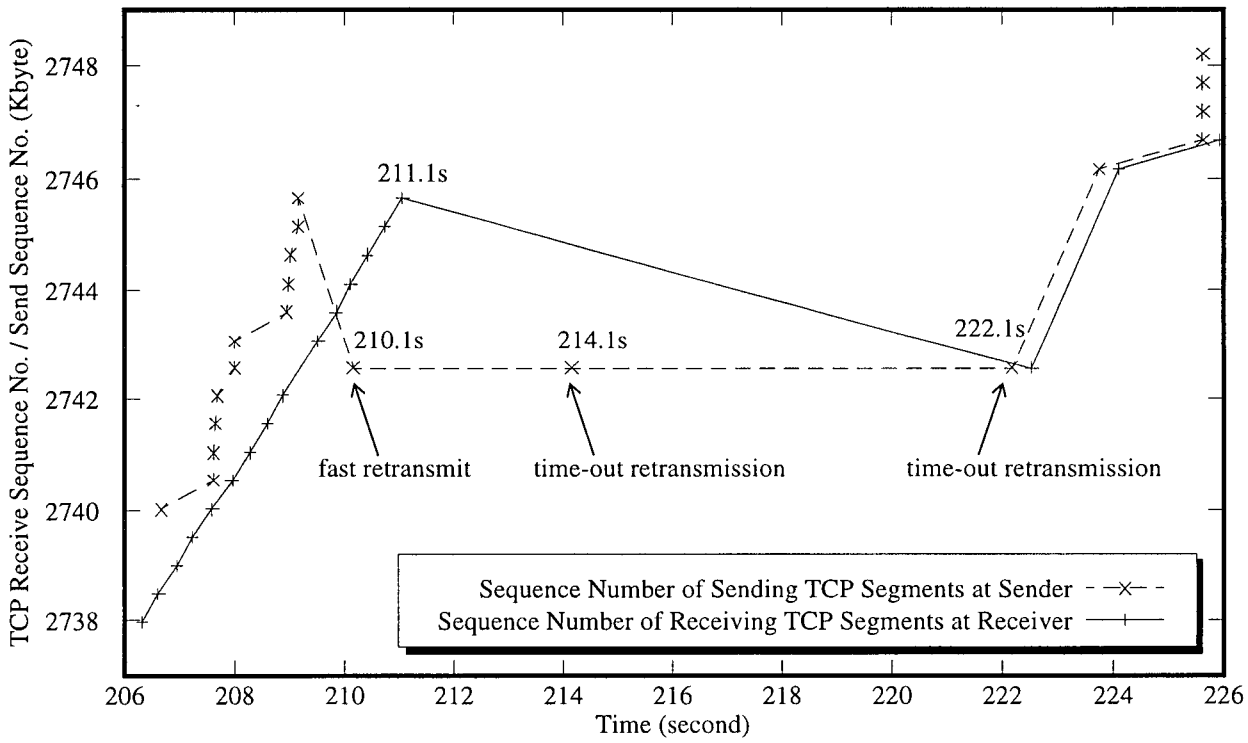

Figure 4.4 Simulation sample of TCP when two consecutive time-outs occur

In summary, wireless losses produce frequent TCP time-outs and invocations of fast retransmit and fast recovery, which reduce the congestion window and can in turn cause more time-outs. Consequently, time-out introduces large end-to-end delay and low throughput performance in TCP. Worst of all, due to the exponential retransmit timer back-off scheme, consecutive time-outs could lead to unacceptably long delay and cause the TCP connection to be idle for a long period of time.

4.2 Behavior of TCP with link layer recovery over the wireless link

The behavior of TCP without link layer recovery is compared with TCP using RLP (section 3.2.3.3) for link layer recovery by simulations at a wireless link PER of 1%.

The dynamics of the TCP congestion windows for both cases are shown in Figure 4.5. Without link layer recovery, frequent packet losses over the wireless link cause TCP to reduce the

congestion window by the combined slow start and congestion avoidance algorithm (as explained in section 2.1.4). As PER increases, the average size of the TCP transmit window decreases due to more frequent reductions in the congestion window, resulting in higher end-to-end delay of TCP and lower TCP throughput. In contrast, the congestion window of TCP with link layer recovery increases to and stays at the maximum value of 4096 bytes, indicating that when PER is low, the link layer recovers almost all wireless losses, effectively preventing time-out retransmissions from occurring at the TCP sender.

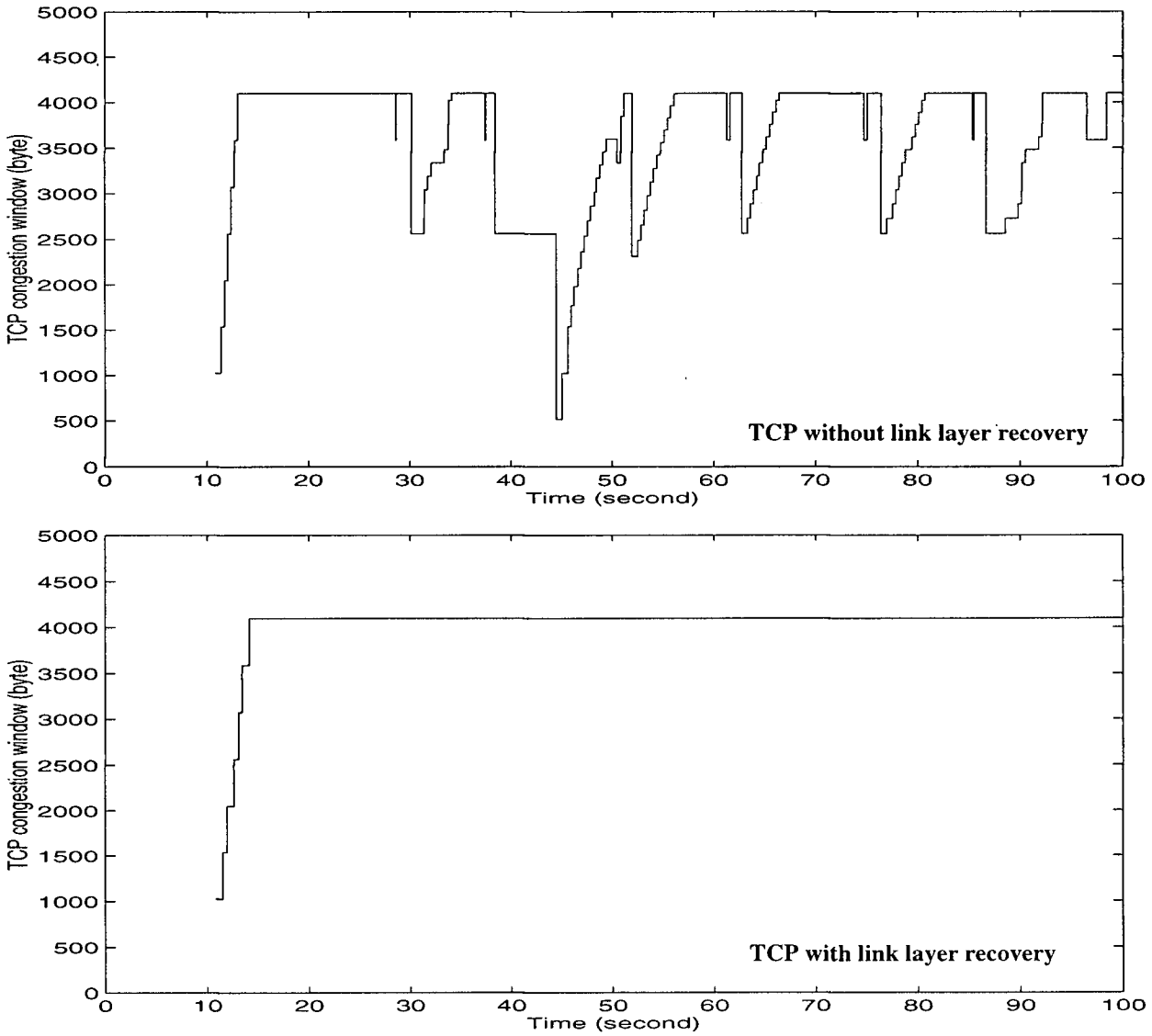

Figure 4.5 Congestion windows of TCP with and without link layer recovery

The traces of the TCP receive sequence numbers are shown in Figure 4.6 for both cases. The higher the rate at which the sequence numbers increases, the higher is the TCP throughput. The traces show that TCP with link layer recovery gives better TCP throughput than TCP without link layer recovery. Without link layer recovery, the **flat** portions of the trace represent periods of time when the TCP receiver is not receiving any segment because the sender is recovering a loss by several successive time-out retransmissions. However, TCP does not suffer from this delay when RLP is used to recover losses over the wireless link quickly.

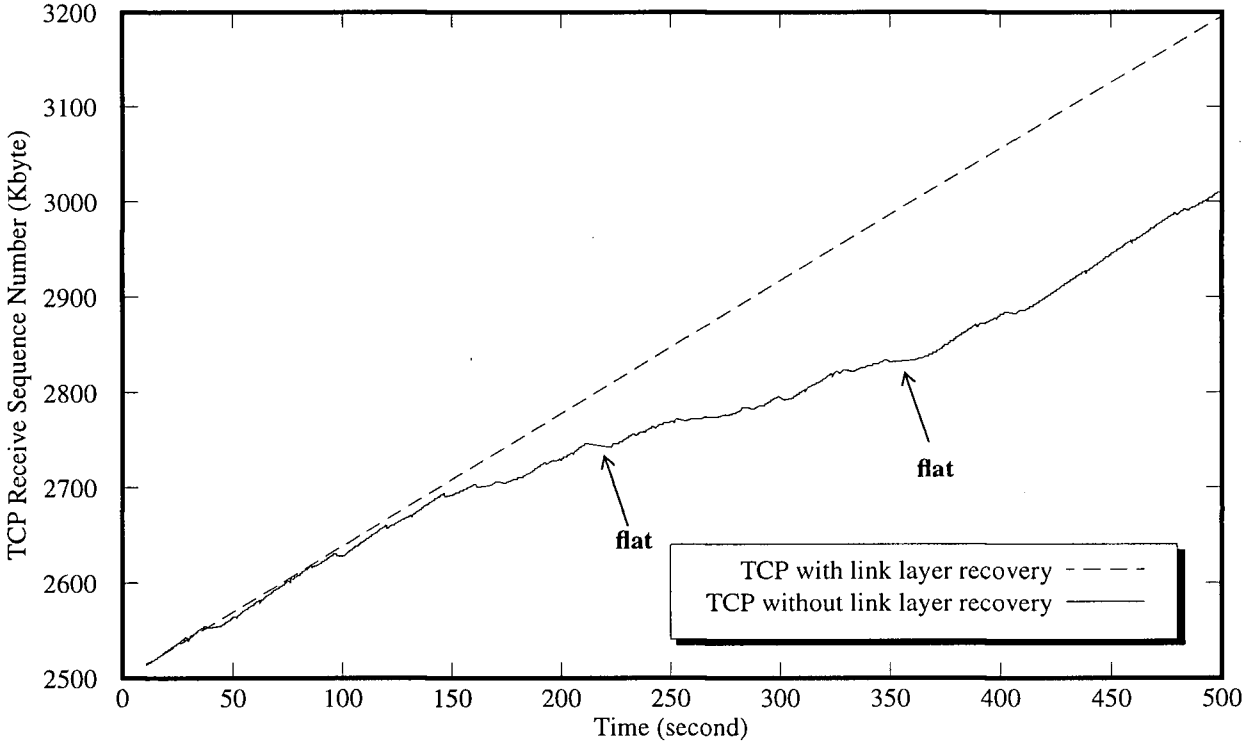

Figure 4.6 Sequence number of receiving TCP segments with and without link layer recovery

Figure 4.7 shows that without link layer recovery, TCP throughput drops rapidly as PER increases since loss of TCP segments reduces the congestion window and slows down TCP transmissions. Worse, the sender times out more frequently, giving the steep increase in average end-to-end TCP delay shown in Figure 4.8. When PER rises to above 10%, average end-to-end delay of TCP increases to more than 5 seconds, causing unacceptable interactive delay to the TCP

application. On the other hand, when link layer recovery is employed, Figure 4.7 and Figure 4.8 show that TCP throughput decreases approximately linearly with PER and the average end-to-end TCP delay increases at a much slower rate.

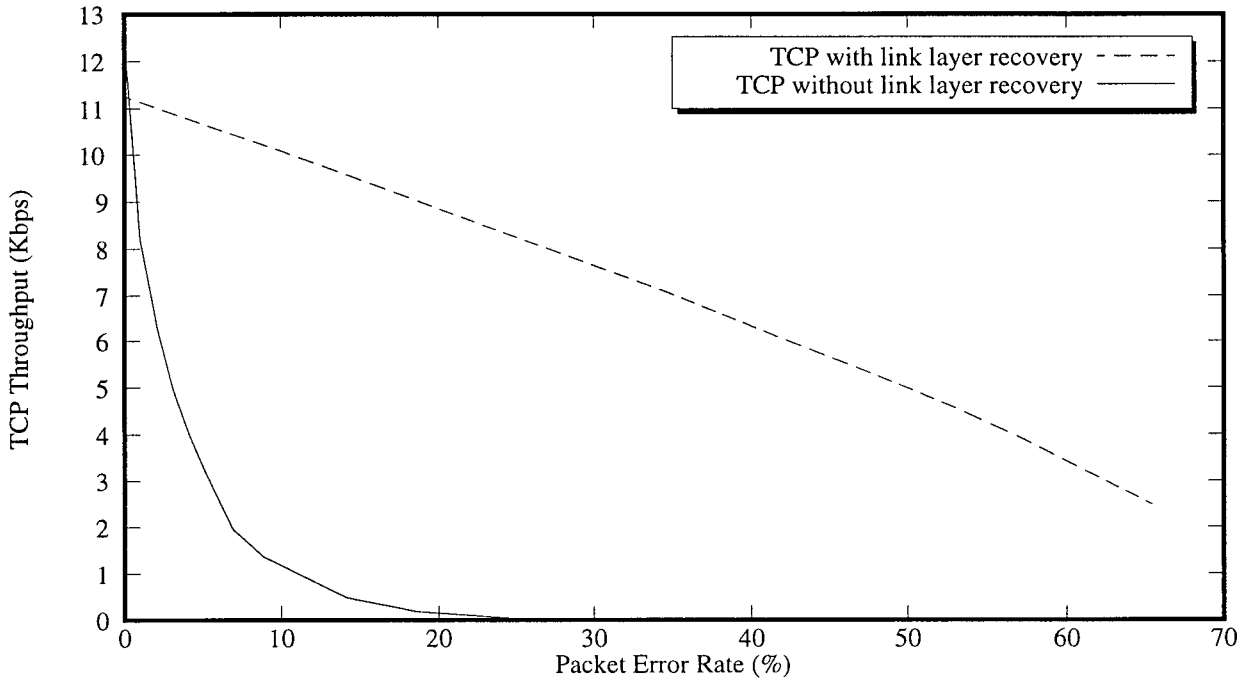

Figure 4.7 Throughput of TCP with and without link layer recovery versus PER

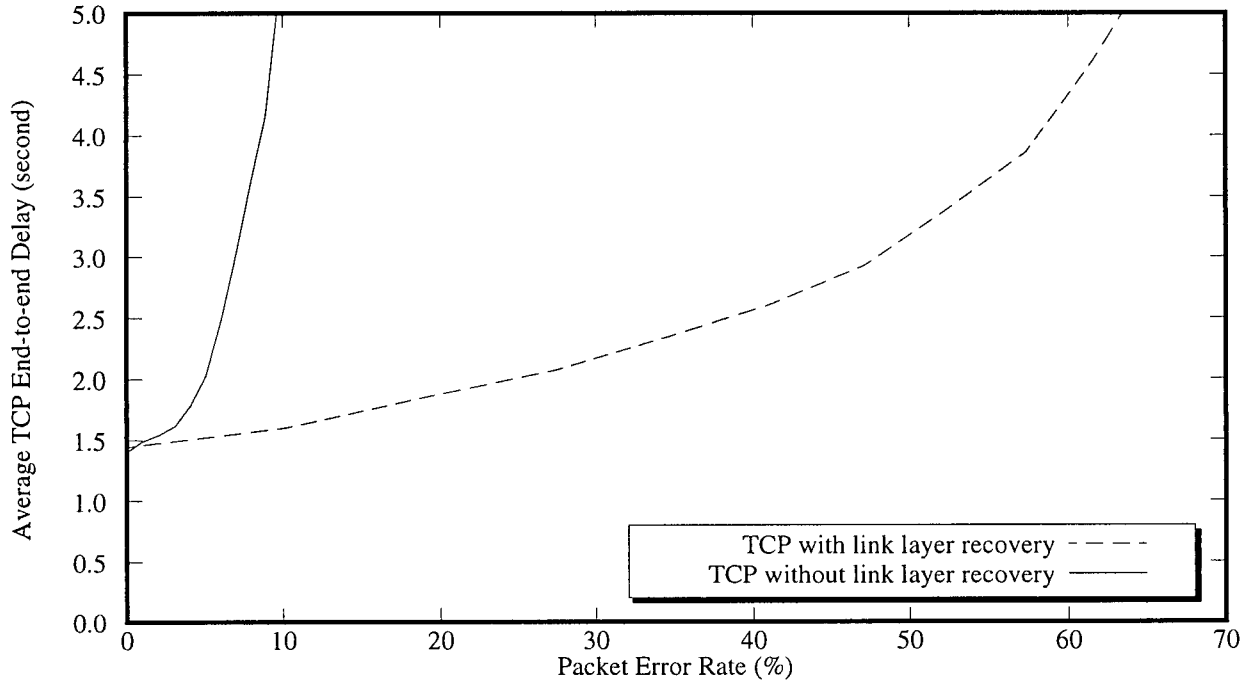

Figure 4.8 Average end-to-end delay of TCP with and without link layer recovery versus PER

The remaining part of this section presents how link layer retransmissions can shield TCP from wireless losses, by showing the effects of link layer retransmission on TCP round trip time estimations and by examining the possibility of competing retransmissions between TCP and link layer. The performance evaluation parameter for the link layer efficiency is the average transmission delay of the data link layer. Figure 4.9 shows that the average transmission delay of RLP rises with PER and the rate of the rise also increases with PER as an increasing number of link layer retransmissions is needed to successfully deliver a link layer frame.

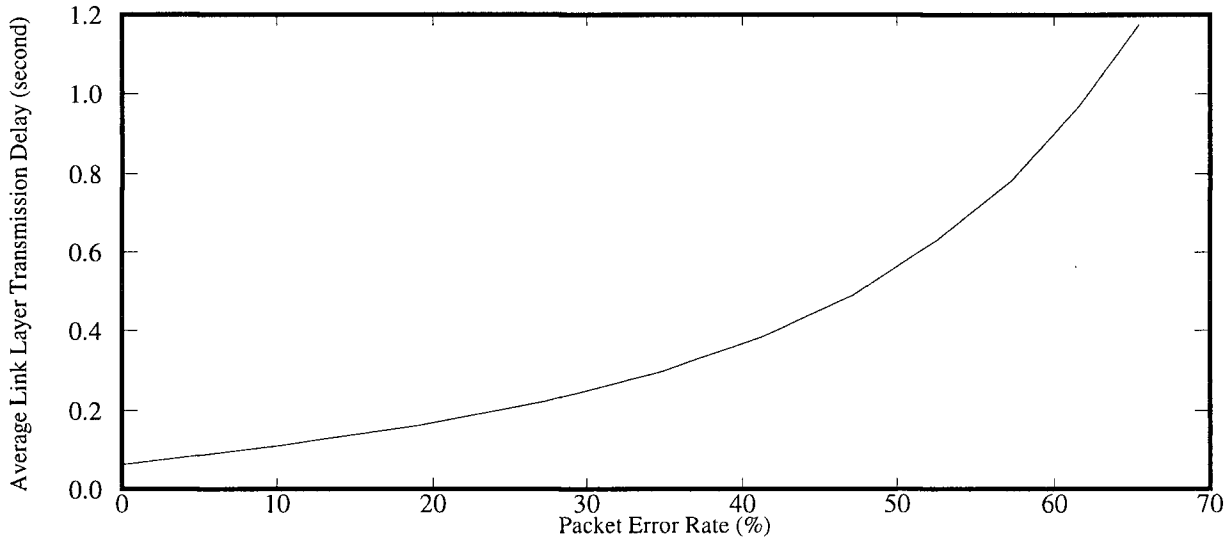

Figure 4.9 Average transmission delay of data link layer versus PER

A possible problem is that the TCP retransmit timer may not adapt well to the link layer recovery delay when PER is high and times out before the link layer recovers a loss, so that both TCP and link layer retransmits the lost data, resulting in unnecessarily duplicated transmissions and loss of efficiency. To investigate the cause of competing retransmissions between TCP and link layer, the effect of link layer recovery on round trip time estimations in TCP (as explained in 2.1.3) is studied. TCP estimates RTT via a running average of the measured time for a segment to be acknowledged. The mean deviation from the average RTT is also recorded. The RTO value is set by the sum of the smoothed RTT and four times its means deviation (Eqn (2.3) in 2.1.3).

In Figure 4.10, without link layer recovery, the average RTT decreases with PER because the RTT is given by the two-way transmission delay of TCP segment and ACK, which decreases when the TCP transmit windows and congestion windows shrink as PER increases, down to a minimum constant value at high PER, when the congestion windows drop to almost one MSS all the time. Note that RTT estimates are updated only by ACKs of those segments that have been transmitted once (Karn's algorithm) [15]. If Karn's algorithm was not employed, the average RTT would be higher as the measurements would include ACKs for segments which have been retransmitted a few times, and RTT would increase with PER as an increasing number of TCP retransmissions is needed to successfully deliver a segment. In contrary, with link layer recovery, average RTT increases steadily with PER as the link layer on average takes a longer time for error recovery before delivering the TCP segments and ACKs in either directions.

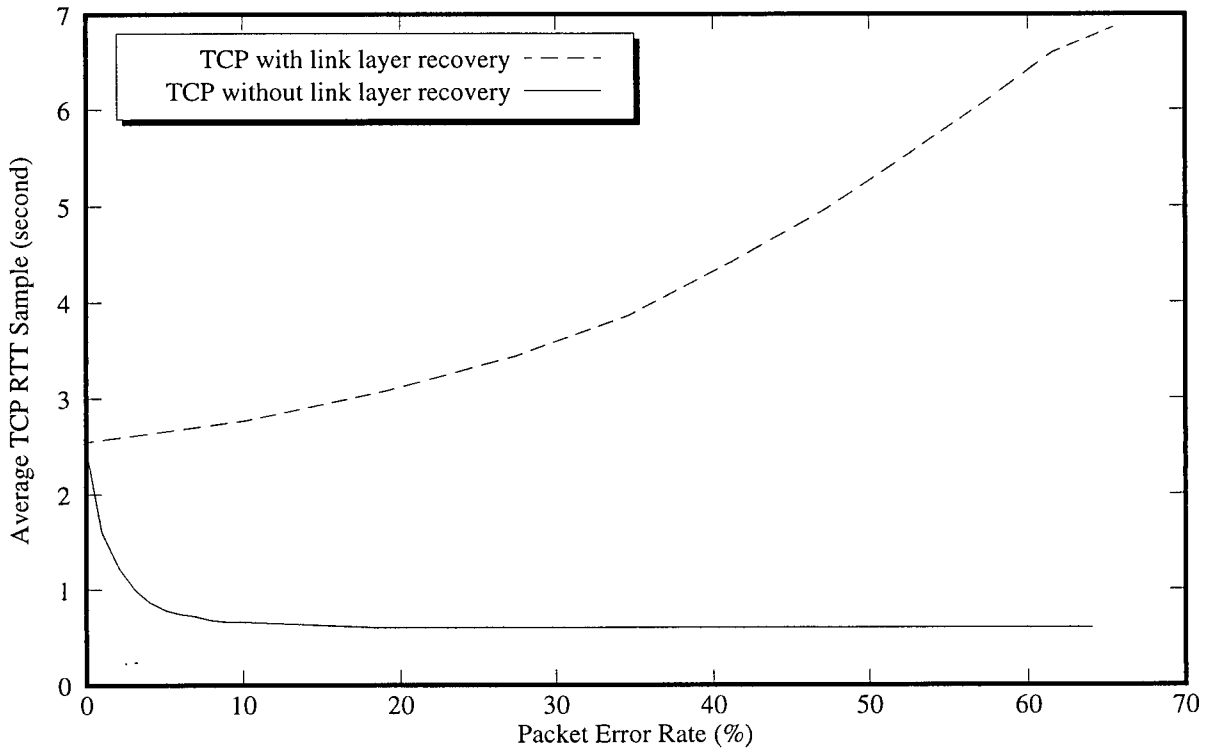

Figure 4.10 Average value of RTT in TCP with and without link layer recovery versus PER

Figure 4.11 shows that the average RTO exhibits the same behavior as RTT, but increases at a faster rate than the RTT estimate in the case of TCP with link layer recovery, due the variability of the RTT samples (i.e., mean deviation of measured RTT also increases with PER).

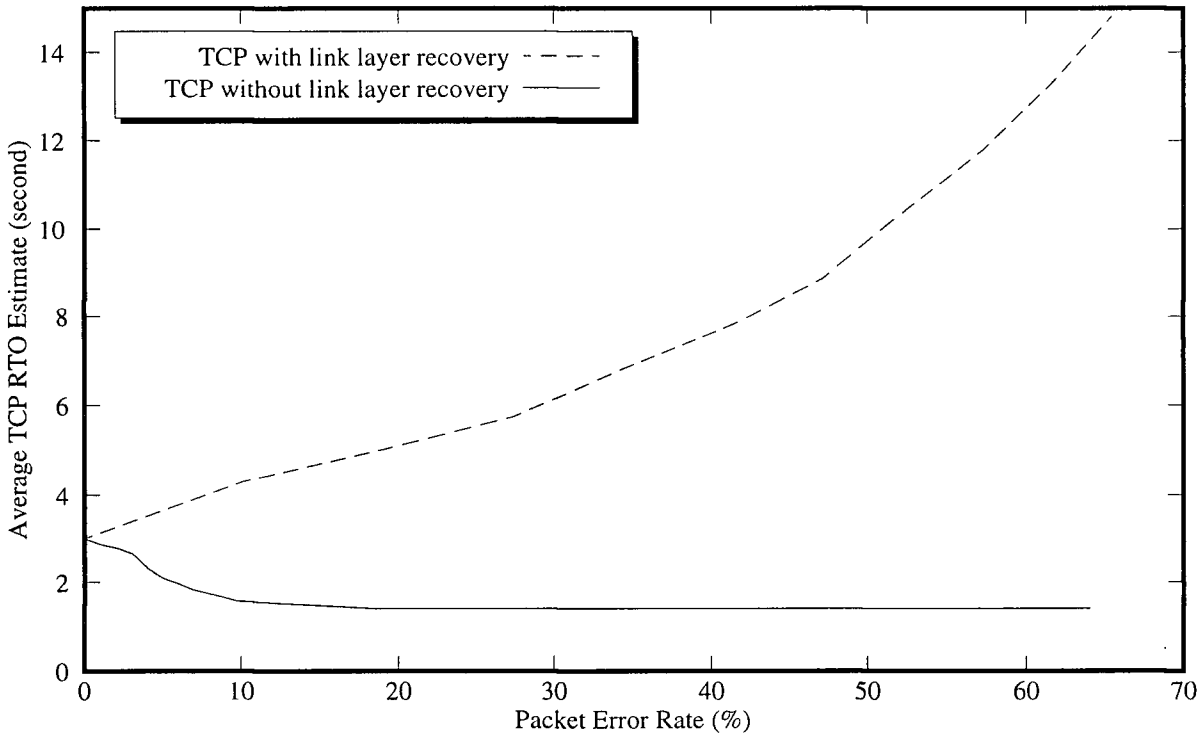

Figure 4.11 Average value of RTO in TCP with and without link layer recovery versus PER

It is apparent that link layer retransmissions improve the throughput and end-to-end delay performance of TCP by shielding it from wireless losses as much as possible. What is seen by TCP is not the lossy wireless link but a more reliable link with a longer and more variable delay. One possible problem is that TCP may not adapt well to the longer and more variable delay and initiate many retransmissions after timing out, before the link layer is able to recover a packet loss. In [11], a serious problem of competing retransmissions between TCP and link layer was reported. However, this problem is not obvious in the mobile data network with slow speed data link considered here, because TCP is able to adjust to the delay by increasing RTO, which reduces

the likelihood of a premature TCP time-out and competing TCP retransmission. The major differences between the simulation model of this thesis and the one employed in [11] are as follows.

- A much lower data rate (19.2Kbps) typical for a mobile data network is considered, compared with the much higher data rate used in [11] typical for wireless LANs.
- A highly effective selective-reject ARQ is used in the link layer retransmission scheme, while a very inefficient stop-and-wait protocol is used in [11].
- The RTT Deviation Coefficient [14] used is 4 instead of 2. Using a higher RTT Deviation Coefficient gives a higher RTO estimate and makes TCP time-outs less likely.

4.3 Effects of link layer re-sequencing on TCP performance

Link layer error recovery employing selective-reject ARQ can be designed to preserve or not to preserve the sequence of packets delivered from the sender to the receiver. For in-sequence packet delivery, the link layer needs a re-sequencing buffer to store any out-of-sequence packets at the receiver. If out-of-sequence delivery is allowed, the link layer may forward any packet received directly to the higher layer and thus no re-sequencing buffer is needed. The effects of these two approaches to link layer recovery on TCP performance are studied in this section, taking into account of packet fragmentations at the IP layer to limit the size of data frames transmitted by the link layer (as mentioned in 3.3). The trade-offs between these two approaches are also mentioned.

For a link layer with in-sequence packet delivery, IP fragments may be stored in the link layer re-sequencing buffer for an extended period of time awaiting recovery of earlier fragments to restore packet sequencing, before the IP fragments can be forwarded to the IP layer. The

average re-sequencing delay generally increases with the PER of the wireless link as link layer recovery takes longer. Figure 4.12 shows that, with in-sequence link layer delivery, TCP waits for long periods of time before receiving a large series of TCP segments. In particular, at 863.2s, 869.8s, 874.2s and 874.8s, 5, 4, 4 and 3 segments are received, respectively.

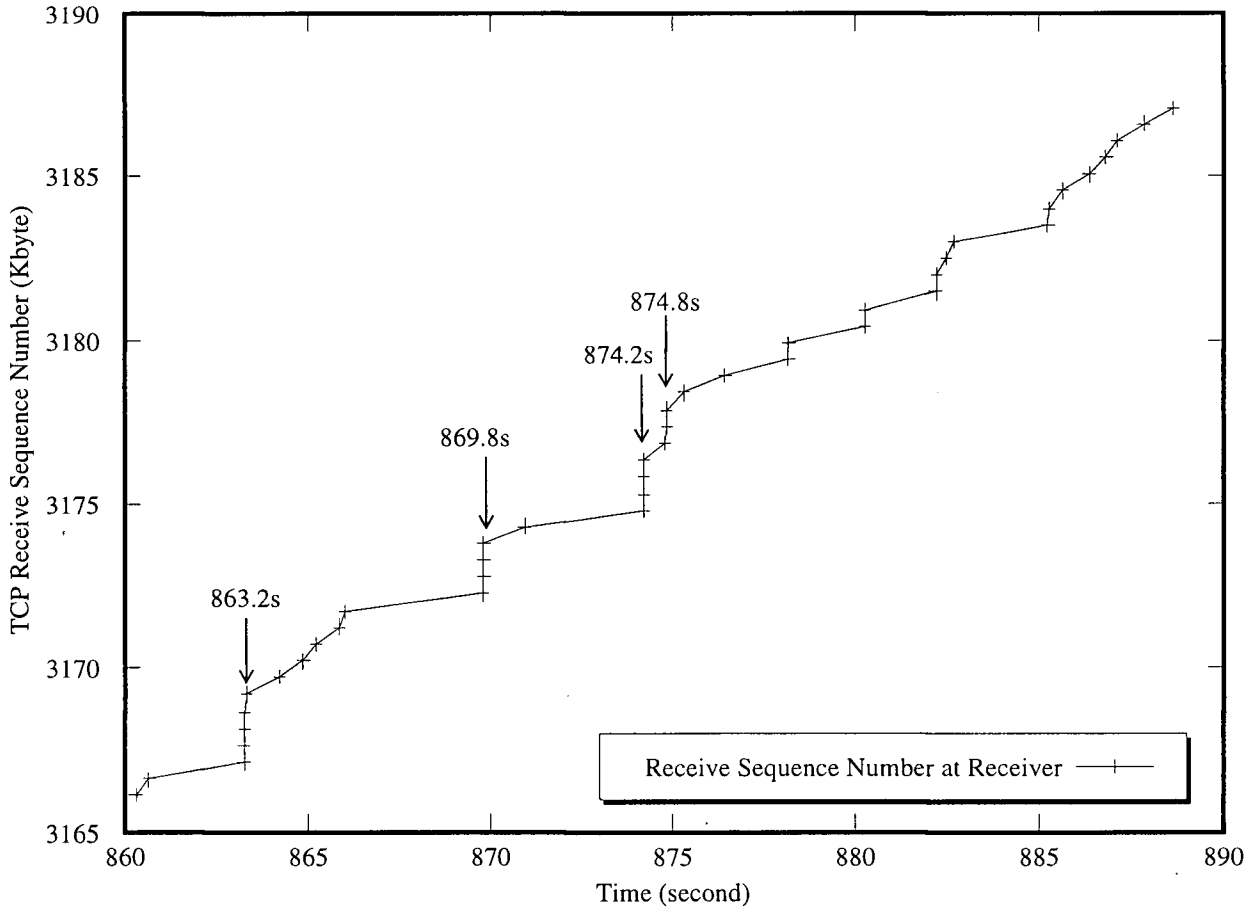

Figure 4.12 Sequence number of receiving TCP segments with in-sequence delivery at data link layer

If out-of-sequence packet delivery is allowed, no re-sequencing delay is involved, and the link layer forwards IP fragments to the IP layer without delay. At the destination host, however, the IP fragments must be reassembled into IP datagrams so that the embedded TCP segments can be delivered to the TCP layer. If several out-of-sequence IP fragments are sufficient to reassemble an IP datagram, TCP will receive a segment out-of-order. This is acceptable since TCP assumes

an unreliable lower layer, and is capable of re-sequencing segments. Figure 4.13 shows that, without in-sequence delivery at link layer, TCP does not suffer the long delay of sequencing link layer but receives segments out-of-sequence.

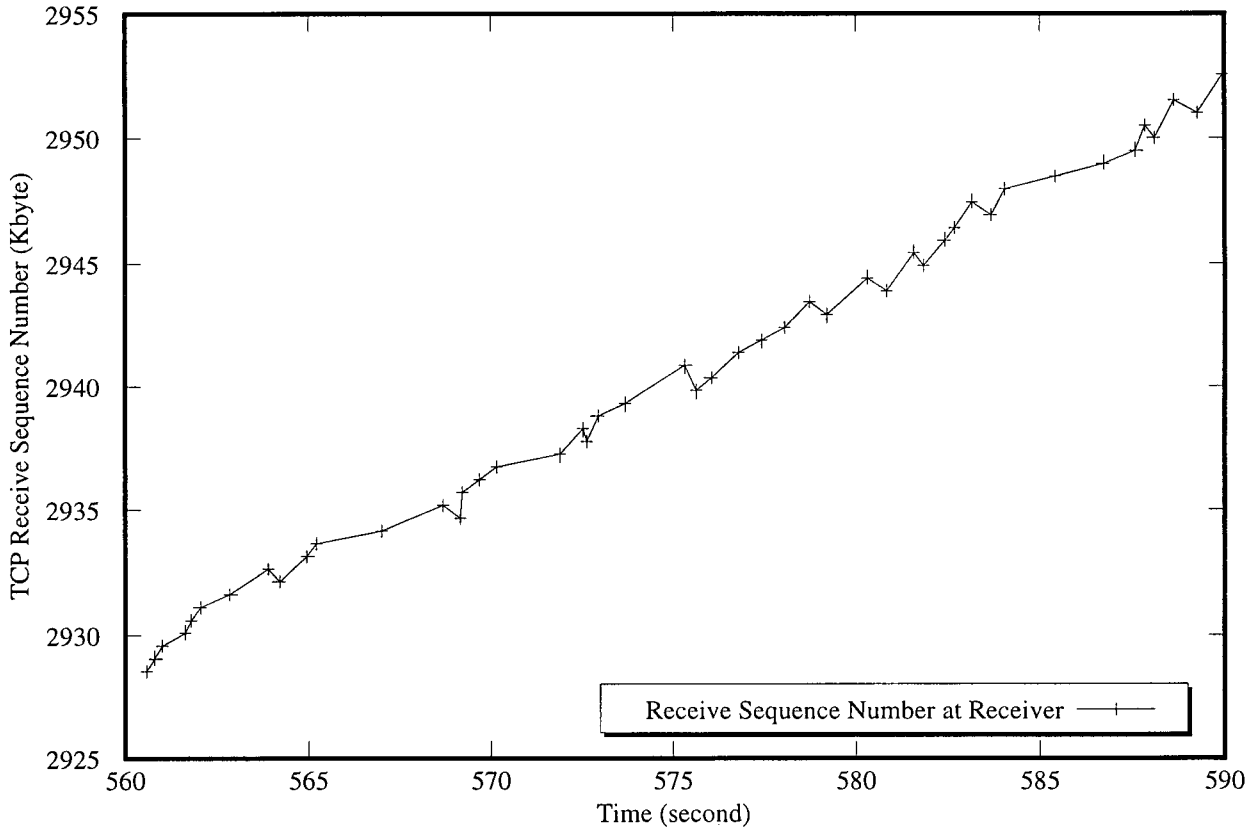

Figure 4.13 Sequence number of receiving TCP segments without in-sequence delivery at data link layer

On the receipt of each out-of-sequence segment, TCP returns a duplicate ACK. Reception of three consecutive duplicate ACKs by the TCP sender causes the sender to invoke fast recovery, decreasing the congestion window and hence reducing the TCP throughput.

In [31], the problem of a non-sequencing link layer is identified, without taking into account of IP datagram fragmentations before transmissions over a wireless link. If the number of fragments per datagram is not too small, e.g., 5 in the simulation model (see section 3.3), *reassembly of IP datagrams provides partial re-sequencing of IP fragments*. To receive an out-of-

sequence TCP segment, all constituent IP fragments need to be received earlier than the lost IP fragment. If the link layer delay is sufficiently small and the protocol ensures that a lost packet is retransmitted at least once before several subsequent packets are sent, the loss will likely be recovered before the destination IP layer reassembles and sends an out-of-sequence segment to the TCP layer.

On the other hand, re-sequencing of IP fragments at the link layer for in-sequence delivery may substantially delay TCP segments and ACKs. At best, delaying ACKs slows the growth of the transmit window at the TCP sender receiving these ACKs, thus limiting increases in TCP throughput. At worst, delaying ACKs for too long may cause the TCP sender to time out, shrinking the congestion window and reducing throughput. Allowing out-of-sequence delivery, however, enables faster return of TCP ACKs, which helps to open the transmit window and prevent time-out retransmissions.

Figure 4.14 shows that TCP without in-sequence delivery at link layer gives a smaller average value of RTT since the transmission delay of TCP segments and ACKs is smaller if a re-sequencing delay is not involved in the link layer. The difference between the two cases increases with PER because the transmission delay of TCP segments and ACKs increases more quickly with PER in TCP with in-sequence delivery.

The above discussion suggests that in-sequence delivery at the link layer does not necessarily give better TCP performance, especially when PER is high. Figure 4.15 compares the TCP throughput using sequencing RLP with TCP using non-sequencing RLP. In the worst case, the non-sequencing RLP has a throughput degradation of only 2.5% below the re-sequencing RLP. In most cases, the TCP throughput for both cases are comparable.

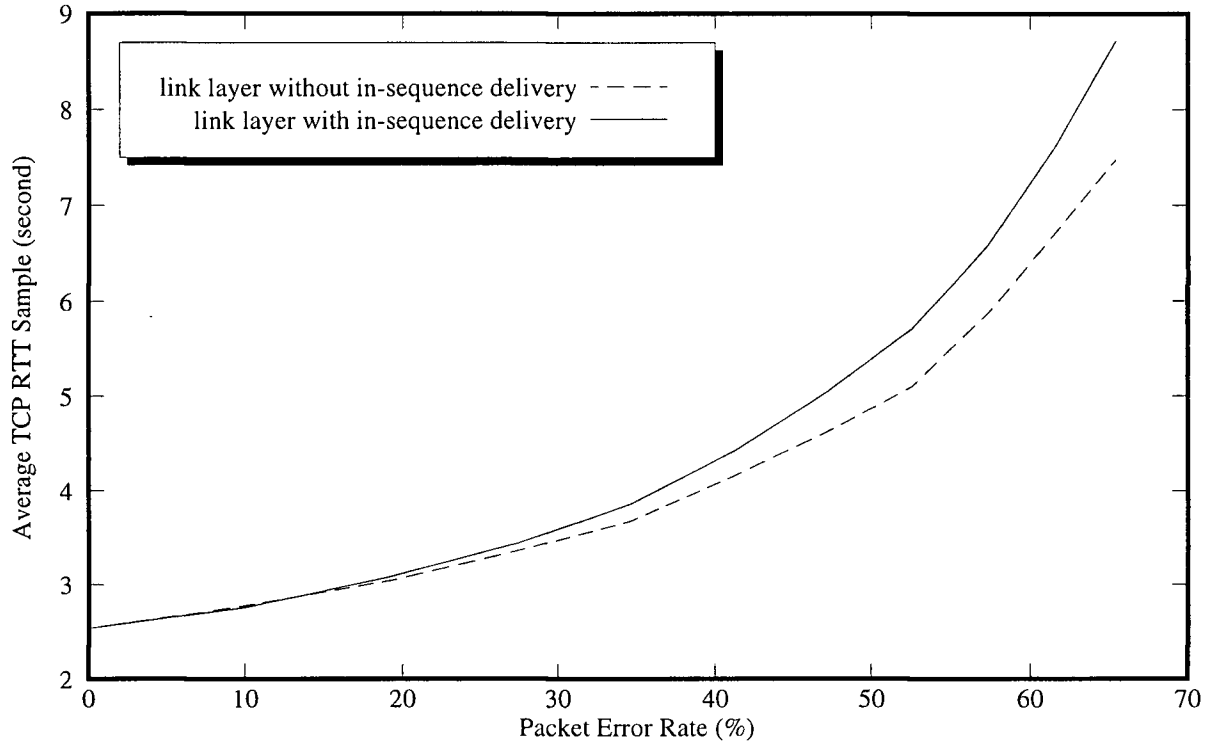

Figure 4.14 Average value of RTT in TCP with and without in-sequence delivery at data link layer versus PER

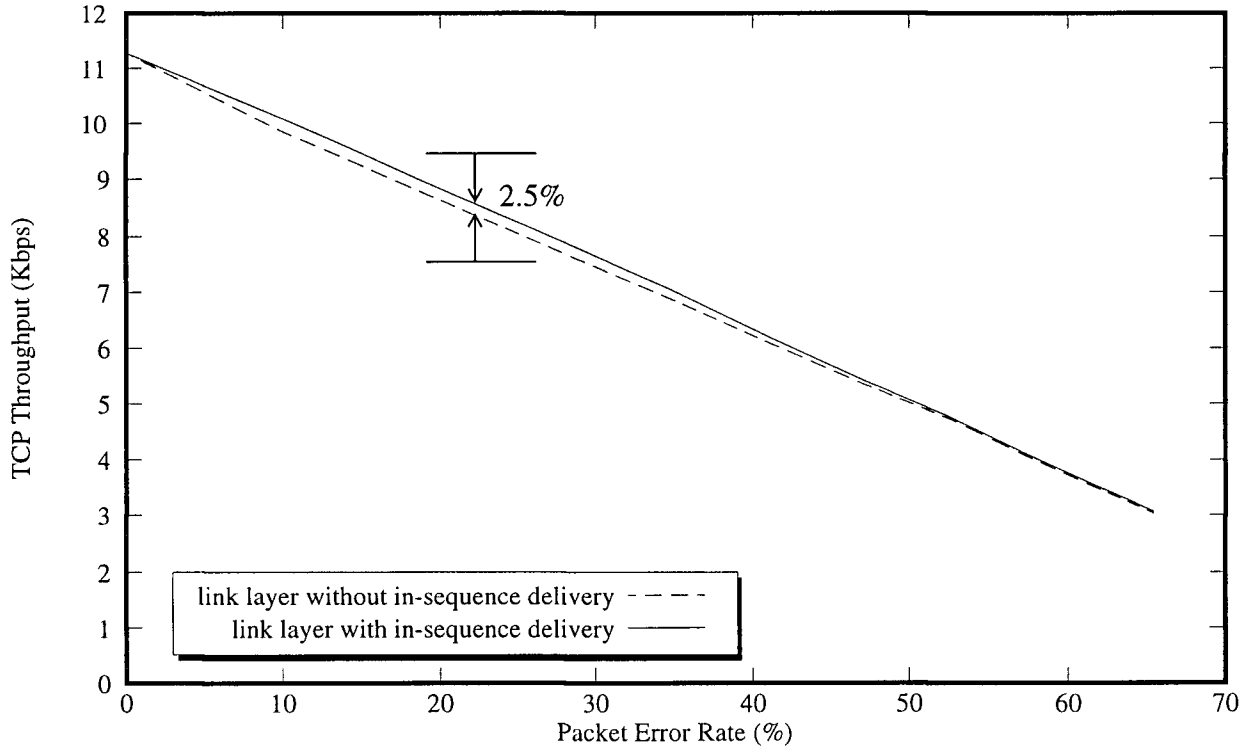

Figure 4.15 Throughput of TCP with and without in-sequence delivery at RLP versus PER

Figure 4.16 compares the TCP throughput using sequencing MDLP with TCP using non-sequencing MDLP. With low PER, non-sequencing MDLP has a slight throughput degradation below the re-sequencing MDLP. However, with high PER, non-sequencing MDLP performs better than re-sequencing MDLP.

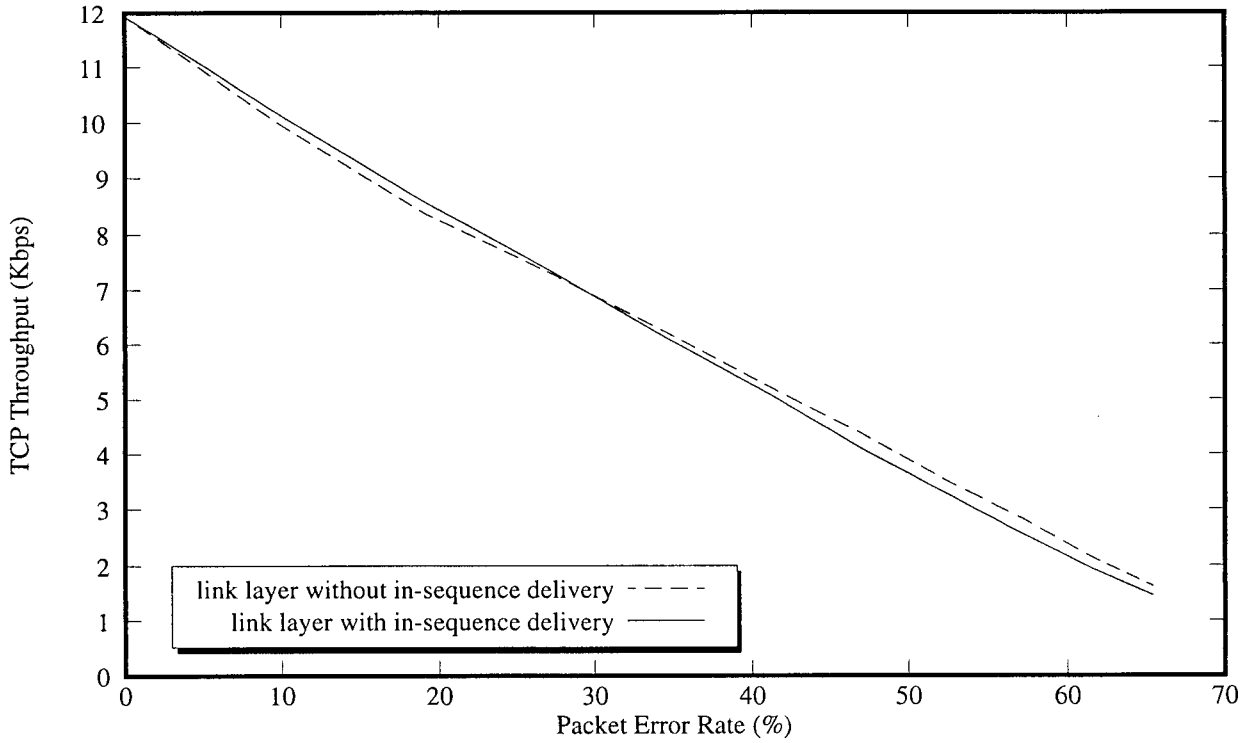

Figure 4.16 Throughput of TCP with and without in-sequence delivery at MDLP versus PER

The trade-offs in choosing between sequencing and non-sequencing link layer protocols depend on the availability of re-sequencing buffer, the computational power of mobile hosts and the number of fragments per IP datagram. For TCP/IP traffic over wireless networks, a *non-sequencing link layer protocol is preferred* because it does not require re-sequencing buffers, thus simplifying the implementation of the data link protocol at both the base station and mobile host.

4.4 Performance of TCP with LAPB, MDLP and RLP for loss recovery

In this section, the performance of TCP using LAPB, MDLP and RLP (section 3.2.3) for loss recovery over the wireless link is compared. Parameters in these three data link protocols that can affect TCP performance are optimized.

4.4.1 Comparison of TCP Performance

Figure 4.17 compares the TCP throughput with different data link protocols (LAPB, MDLP and RLP). Except when PER is less than about 10%, RLP performs better than MDLP and LAPB. When compared with RLP and MDLP, TCP throughput with LAPB decreases much faster with increasing PER.

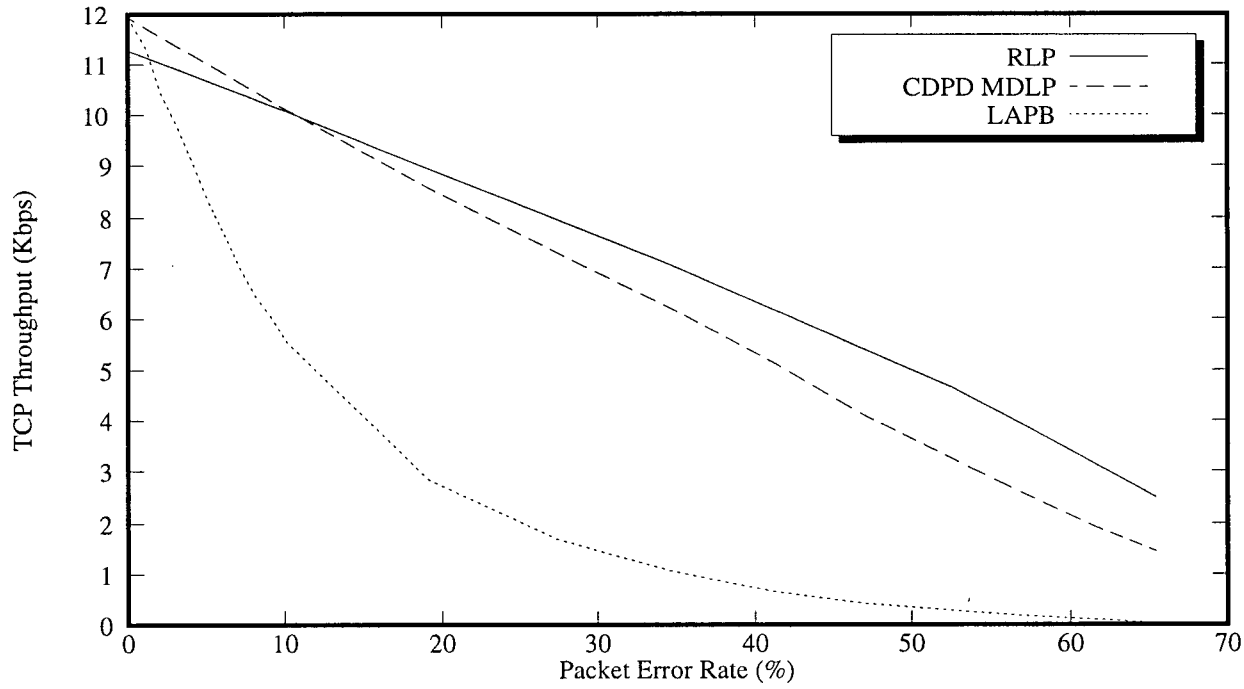

Figure 4.17 Throughput of TCP using LAPB, CDPD MDLP and RLP for loss recovery

LAPB uses a go-back-N ARQ in which all out-of-sequence frames received are discarded. When PER increases, the number of discarded frames increases and thus much more useful capacity of the wireless link is wasted, resulting in a rapid decrease in TCP throughput.

MDLP uses a selective-reject ARQ in which out-of-sequence frames are not discarded. However, another situation where MDLP may waste the capacity of the wireless link after a time-out retransmission at the MDLP sender is illustrated in Figure 4.18. Normally, the receiver will receive a few number of duplicate frames after a time-out retransmission at the sender because the sender does not have *exact* information on which frame the receiver has not received and therefore it reverts to retransmit all the I frames, causing unnecessary duplications and thus wasting the capacity of the wireless link.

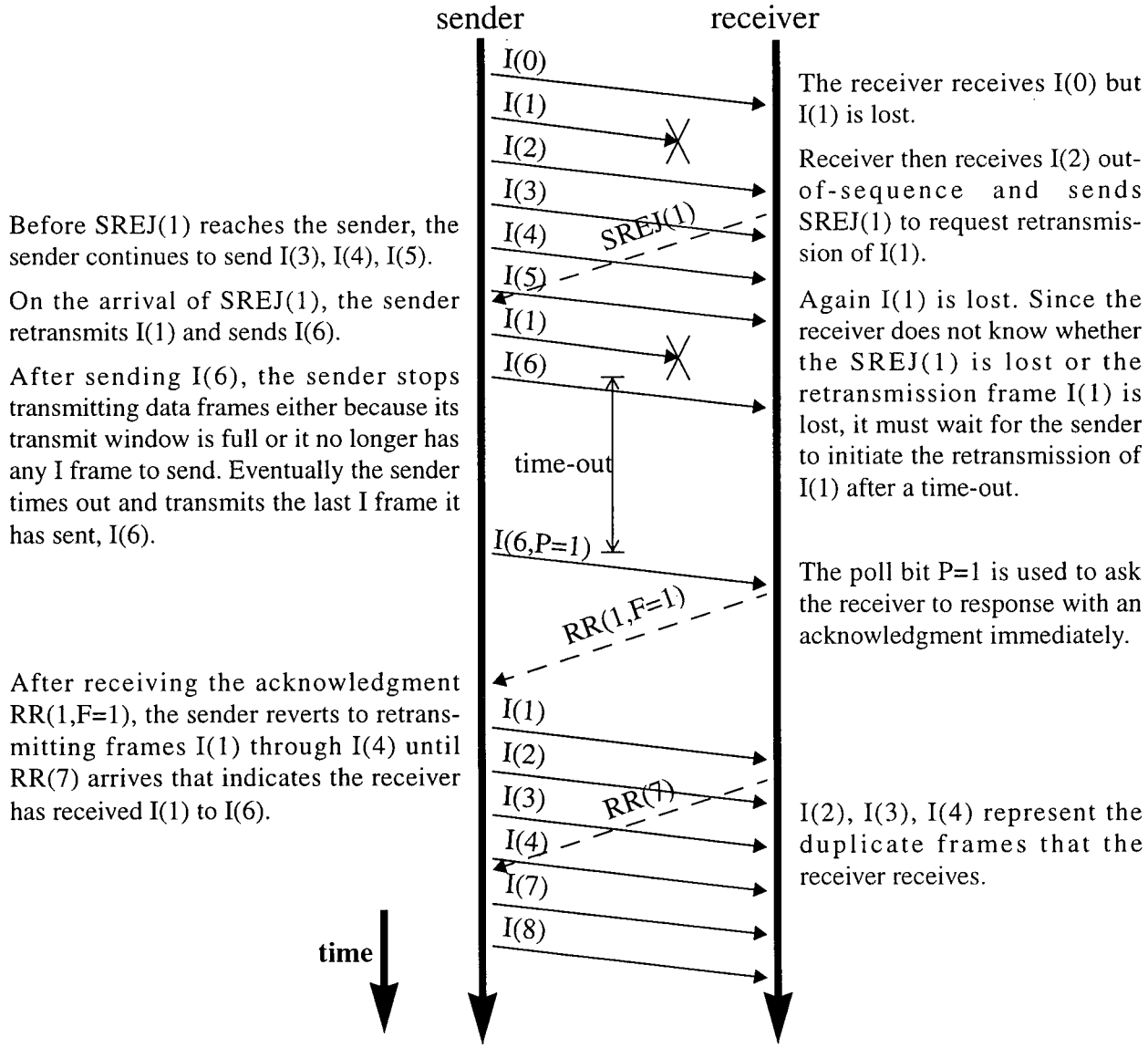

Figure 4.18 Illustration of duplicate retransmissions in MDLP

RLP uses a selective-reject ARQ too. However, the retransmissions in RLP are receiver-initiated only and thus no time-out retransmission occurs at the sender, avoiding the previous problem of MDLP. Moreover in RLP, receiver uses periodic status messages to inform the sender *exactly* which frame is received and which frame is missing. As a result, RLP performs better than MDLP when PER is high. When compared with LAPB and MDLP, RLP spends more wireless capacity on acknowledgments because RLP needs to send periodic status messages even when both ends of the data link connection have no data to send. This results in lower TCP throughput than LAPB and MDLP when PER is low.

In conclusion, data link protocols using a go-back-N retransmission scheme (like LAPB) are not recommended in an error-prone wireless environment because of their inefficiency. Selective-reject, on the other hand, is preferred since today's mobile computers have enough computational power for complex protocol processing and buffer for storing out-of-sequence frames. Over a lossy wireless link, receiver-initiated protocols (like RLP) are favored over sender-initiated protocols (like MDLP) because they do not produce as many duplications as sender-initiated protocols, thus saving the scarce wireless bandwidth.

4.4.2 Parameters Optimization

A common parameter used in these three data link protocols that may affect link recovery performance is the transmit window size, which defines the maximum number of data frames that may be outstanding at any given time. Since the data link layer and MAC layer exchange primitives with each other so that the flow of link layer frames to the MAC layer can be controlled, a large transmit window does not jam the queue at the MAC layer. However, the maximum transmit window size allowable to a *selective-reject* protocol should be less than half the range of sequence numbers [1]. Since a 7-bit sequence number is used for these three data link

protocols [13][28], the maximum transmit window size is 63 (i.e $2^7/2$-1). The suggested value of the transmit window size is 15 in [13]. Simulations show that TCP throughput increases with the transmit window size of data link layer until it reaches about 12, suggesting that the size of 15 is a good choice.

In addition to the transmit window size, RLP has the following two parameters that may affect link recovery performance (section 3.2.3.3):

- period of receiver status feedback timer, and
- minimum time between transmissions.

The period of the receiver status feedback timer controls the time between successive acknowledgments or retransmission requests sent by the receiver. Figure 4.19 shows the effect of the status timer period on TCP throughput. A longer status timer period uses less wireless capacity on ACKs, giving higher TCP throughput when *PER is low*. However, a shorter status timer period enables the receiver to inform the sender about its losses more quickly, thus enabling a faster loss recovery and resulting in higher TCP throughput when *PER is high*. The best value of the status timer period is about 0.2s.

When the RLP sender receives a status message, it does not retransmit all unacknowledged frames but rather retransmits a frame only if the time from the previous transmission of this frame exceeds the *minimum time between transmissions*, so as to ensure that this frame is really lost rather than delayed when the status message is sent by the receiver. Figure 4.20 shows the effect of this parameter on TCP throughput. If the minimum time between retransmissions is too large, sender may take too long to recover a lost frame. On the other hand, if this parameter is too small, sender may retransmit data frames too frequently, causing duplications. The best value of the minimum time between transmissions is found to be about 0.3s.

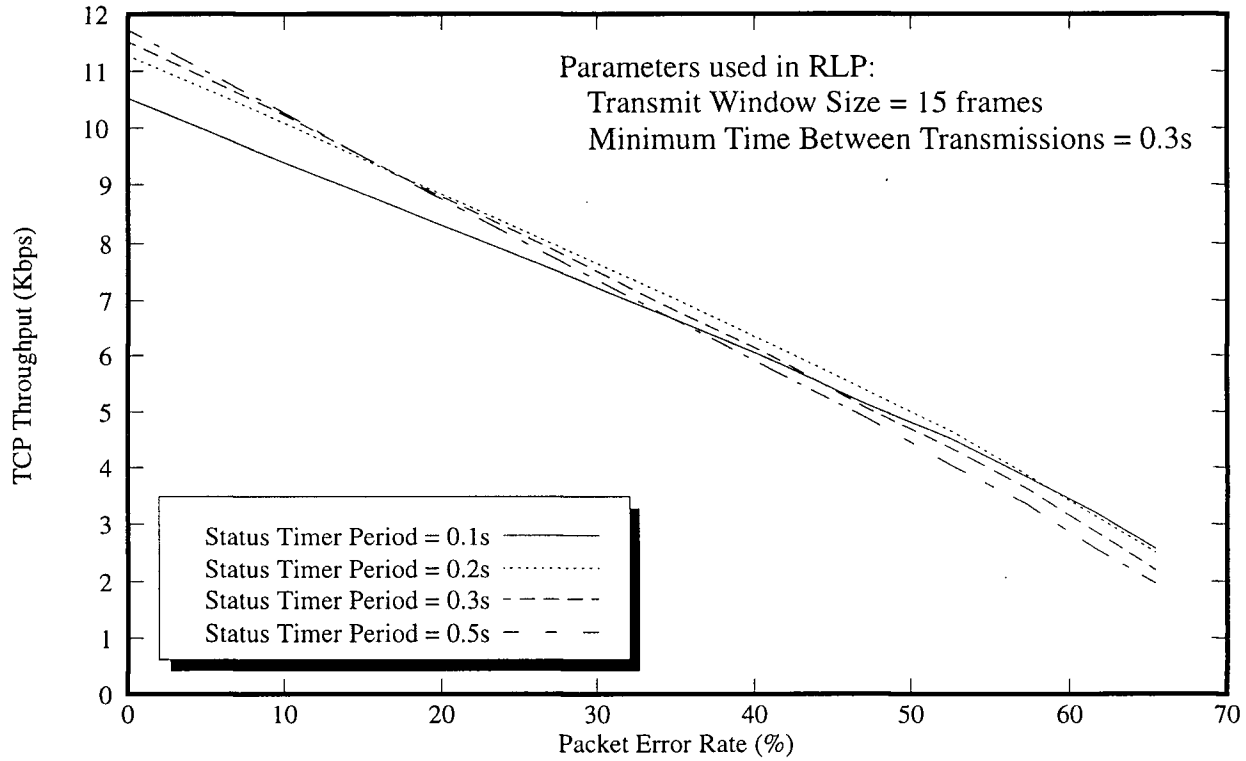

Figure 4.19 Throughput of TCP versus period of receiver status feedback timer (RLP)

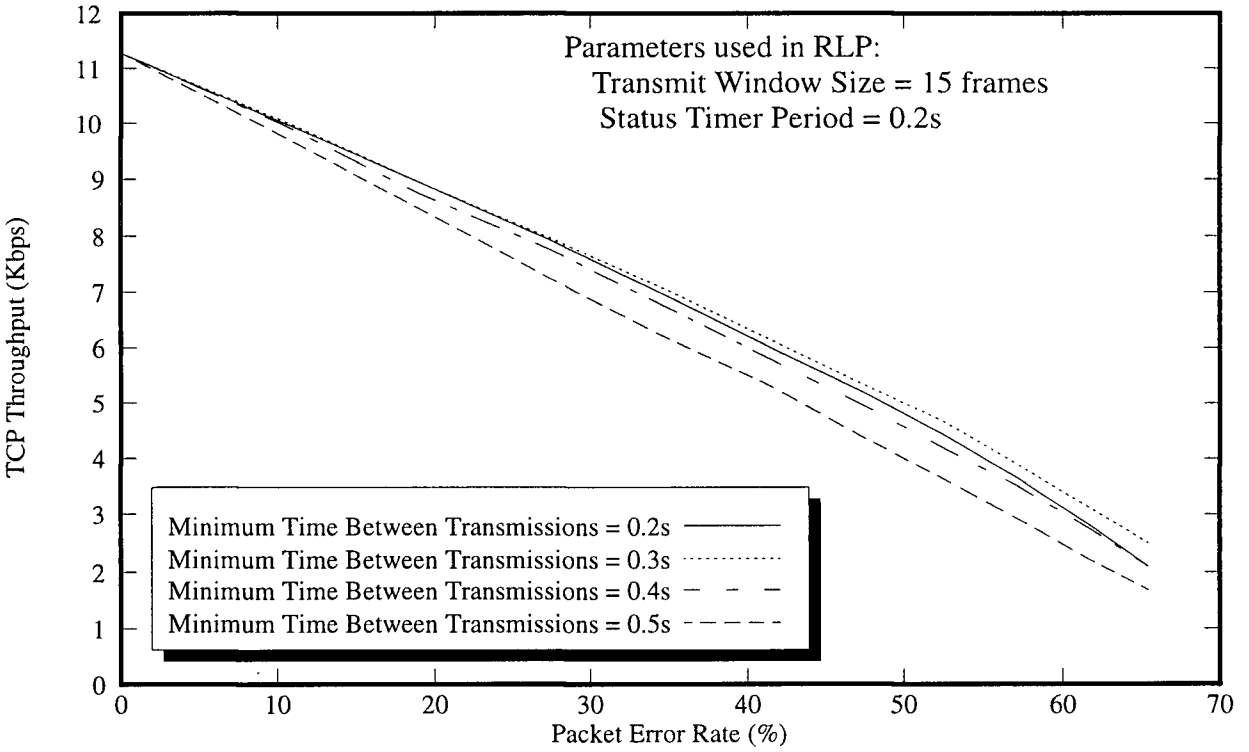

Figure 4.20 Throughput of TCP versus minimum time between transmissions (RLP)

MDLP has the following two parameters that may affect performance (section 3.2.3.2):

- maximum ACK delay (T205 timer time-out value), and
- retransmit timer time-out value (T200 timer time-out value).

Unlike RLP, an MDLP ACK is piggybacked onto an I frame if an I frame is sent before the T205 timer expires. If the maximum ACK delay is too small, most ACKs are not piggybacked onto I frames, thus more transmission capacity is utilized on sending single RR frames. However, if the maximum ACK delay is too large and no I frame is sent onto which an ACK can be piggybacked, the transmission of ACKs may be delayed for too long. [13] suggests the maximum ACK delay to be 0.5s. It is found by simulations that the best value is about 0.3s. Figure 4.21 shows the effect of the time-out value of the MDLP retransmit timer on TCP throughput. A too large time-out value may delay loss recovery but a too small time-out value may produce many duplications. In Figure 4.21, the best choice is about 1s.

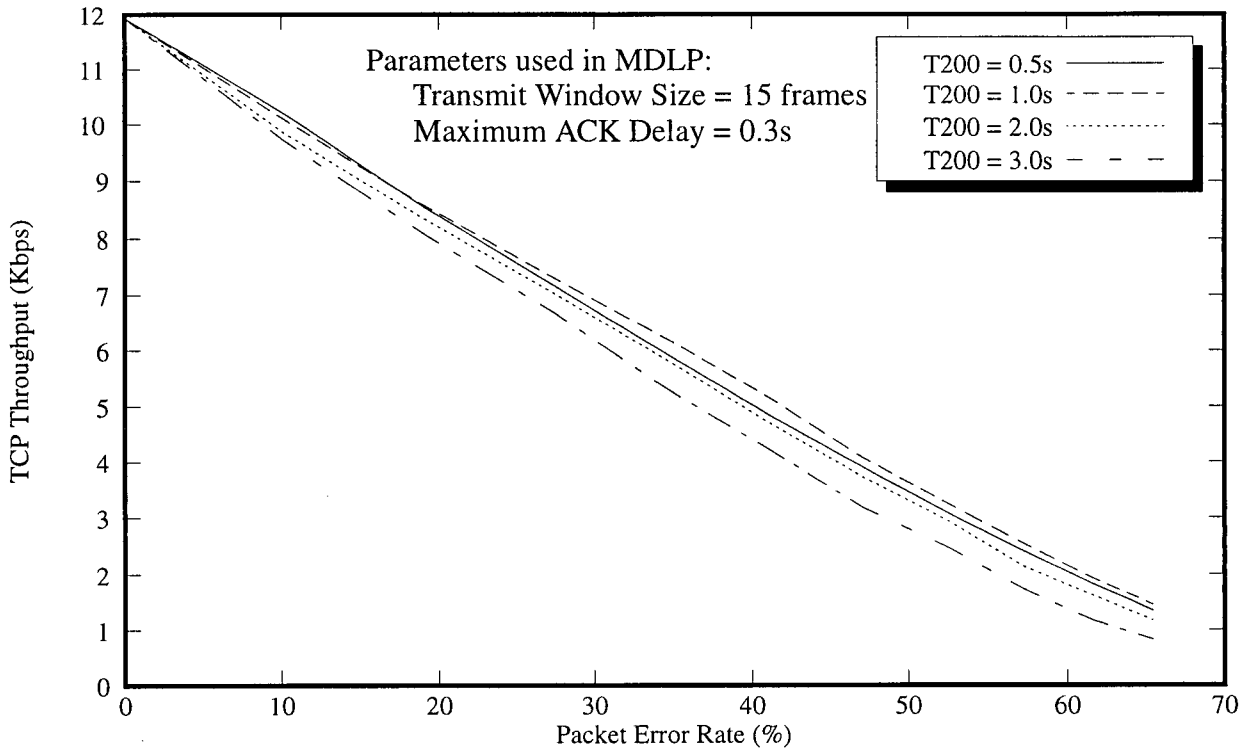

Figure 4.21 Throughput of TCP versus T200 retransmit timer time-out value (MDLP)

LAPB has the following two parameters that may affect performance (section 3.2.3.1):

- maximum ACK delay (T2 timer time-out value), and
- retransmit timer time-out value (T1 timer time-out value).

Similar to MDLP, LAPB enables ACKs to be piggybacked onto I frames. A T2 timer with the same function as the T205 timer in MDLP is used. With the same argument as in MDLP, the best value of the maximum ACK delay is found by simulations to be about 0.3. LAPB uses a T1 timer to trigger a retransmission after a time-out, which functions like the T200 timer in MDLP. The effects of the retransmit timer time-out value on TCP throughput is similar to that in MDLP. In Figure 4.22, the optimum choice for the time-out value is about 1.0s.

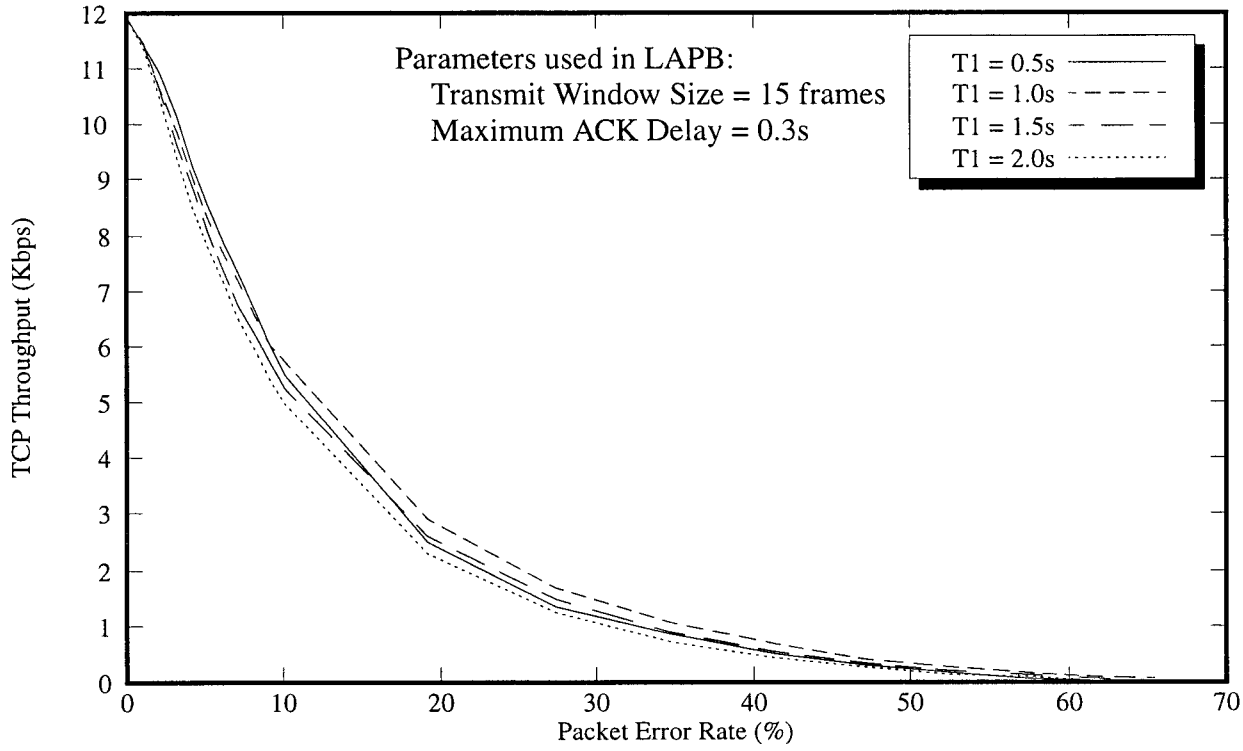

Figure 4.22 Throughput of TCP versus T1 retransmit timer time-out value (LAPB)

Chapter 5 Performance Analysis of TCP with Best-Effort Link Layer Retransmissions

Three types of operation can be defined in the data link layer according to the services it provides to the higher layer: unreliable, reliable and best-effort. The performance analysis of TCP using unreliable and reliable data link protocols was given in Section 4.1 and 4.2 respectively. In this chapter, the best-effort data link protocols are considered. Modifications for the best-effort data link protocols are proposed and performance evaluation of the best-effort approach is given.

5.1 Modifications to data link protocols for best-effort service

Data link protocols like LAPB [28] and MDLP [13] do not ensure *absolute reliability* in data delivery. In fact there is always a trade-off between reliability and delay, i.e., more reliability leads to longer recovery delay and vice versa. In LAPB and MDLP, if a packet loss cannot be recovered after several retransmissions, the data link connection will *reset* and needs to be re-established. A link layer reset causes all data packets queued in the link layer to be discarded and the higher layer to be informed that the connection has been terminated.

In the wireless environment, when the channel condition is bad, there can be frequent link layer resets resulting in loss of TCP segments and significant degradation of TCP performance. Therefore in the thesis, a best-effort approach is proposed, in which the data link layer does most of the recovery for packets lost in the wireless link while TCP does the rest, and the data link connection is maintained all the time without being reset. The suggested modifications for the best-effort data link protocols depend on whether link layer loss recovery is initiated by the sender, the receiver, or both (see Section 3.2.3).

5.1.1 Modifications in sender-initiated and both-initiated data link protocols

If loss recovery is initiated by the sender or both the sender and receiver using time-out retransmissions as in LAPB and MDLP, the sender uses a state variable to keep track of the number of successive time-out retransmissions. The state variable is reset to zero once a link layer ACK is received indicating receipt of the retransmission frame. Each successive time-out increments the state variable by one. If the state variable reaches some threshold value, the **maximum number of retransmissions**, the sender and receiver reset and re-establish a new connection as shown in Figure 5.1 [28][13].

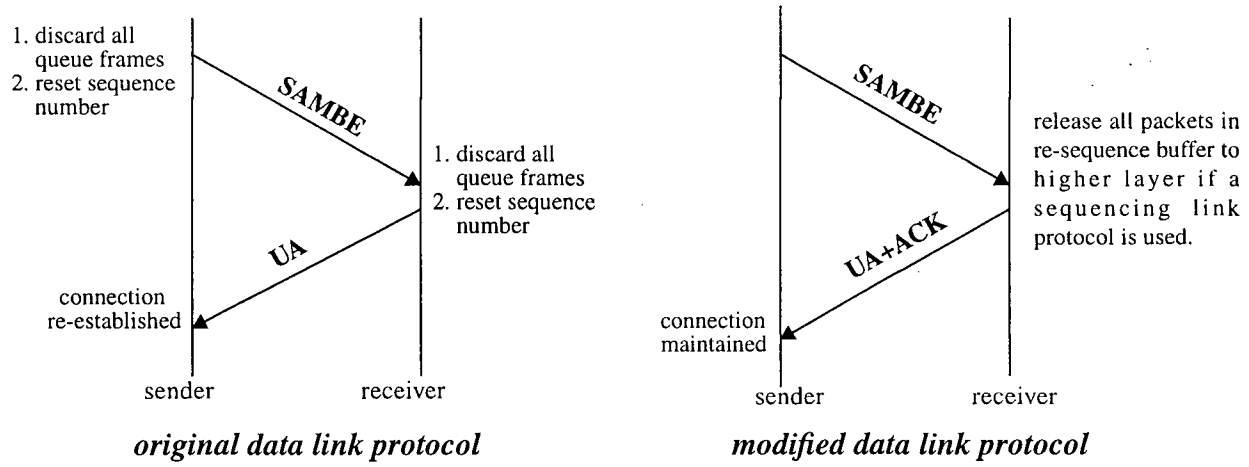

Figure 5.1 Comparison between original and modified data link protocols

In the original data link protocol, during connection re-establishment, the sender transmits a SAMBE command frame to the receiver, discards all queued packets and resets the sequence numbers it has been using. Upon receipt of the command, the receiver returns a UA response frame to confirm the connection request, discards all queued packets and resets its sequence number. When the UA response reaches the sender, a new connection is established.

In the modified data link protocol, the SAMBE command and UA response are exchanged as before. However, queued packets are not discarded and sequence numbers are not reset. Upon

receipts of a SABME, the receiver delivers all packets in the re-sequencing buffer to the higher layer if the link layer protocol attempts in-sequence delivery, advances the receive window, and returns a combined UA+ACK frame indicating the sequence number of the next packet expected in the now empty receive window. When the UA+ACK response reaches the sender, it advances the transmit window to start sending packets at the sequence number specified by the UA+ACK frame, after clearing from the transmit buffer those packets with smaller sequence numbers.

Note that in both protocols, when a SAMBE command or an UA (or UA+ACK) response is lost due to wireless error, the sender will retransmit the SAMBE command after a time-out until it receives the response from the receiver.

5.1.2 Modifications in receiver-initiated data link protocols

If loss recovery is receiver-initiated as in RLP, a different algorithm is needed to modify the data link protocol for best-effort retransmissions. For instance, in RLP, the receiver sends periodic status feedback to request retransmissions from the sender for the losses it detected. Since the sender does not necessarily retransmit for every retransmission request received (see section 3.2.3.3), the receiver has no knowledge about how many times the sender retransmits a certain packet. Therefore receiver cannot use the *number of retransmission requests* for a lost packet to determine whether it should request further retransmission or abandon the request of retransmission. The following modifications to add the best-effort approach to receiver-initiated data link protocols is proposed.

When a missing packet is initially detected, the receiver records the *time* at that instant and the *sequence number* of the packet, and stores the record at the end of a list. When a missing packet is received, the corresponding record is removed from the list. Before the receiver returns a

status message (sent periodically),

(1) it compares the difference between the current time (t_c) and the recorded time of the first missing packet on the list (t_1) against a threshold parameter called **maximum recovery delay** (*mrd*).

(2) If $t_c - t_1 > mrd$, the record for this packet is deleted from the list so that it is not included in the retransmission request of the next status frame. The receiver releases to the upper layer all packets in the re-sequencing buffer with smaller sequence numbers than this packet if the link layer protocol attempts in-sequence delivery and continues to check on the first missing packet in the updated list by repeating step (1).

(3) If $t_c - t_1 \leq mrd$, the list is unchanged and the next status frame can request retransmission of all packets on the list.

5.2 Behavior of TCP with best-effort link layer recovery

In the sender-initiated and both-initiated data link protocols, the **maximum number of retransmissions** controls the reliability of the link layer providing to the higher layer. In the receiver-initiated data link protocols, the **maximum recovery delay** determines the reliability of the link layer. In some data link protocols, the maximum number of retransmissions is recommended in the specifications [13] [28]. However, there was no previous research on the suitable value of the maximum recovery delay. Since these two parameters may affect TCP performance significantly, it is interesting to analyze the trade-off between the degree of reliability in the link layer and the TCP performance.

In Figure 5.2, the effect of N2 (maximum number of retransmissions) on the performance of TCP with unmodified MDLP is compared to that with MDLP modified for best-effort retransmissions. Both the original MDLP and best-effort MDLP do not attempt in-sequence delivery. The TCP throughput of sequencing MDLP is found to be close to that of non-sequencing MDLP (as mentioned in section 4.3) and thus is not shown here. For all three values of N2, the modified MDLP gives better TCP throughput than the original MDLP. When N2 is small, best-effort retransmissions give an improved TCP throughput performance compared to original MDLP with link resets, particularly when PER is high. However, as N2 increases or PER decreases, as expected, the difference in performance diminishes. In particular, the percentage increases in throughput are about 81% with N2=3, 22% with N2=5 and 2% with N2=7, at PER of 50%.

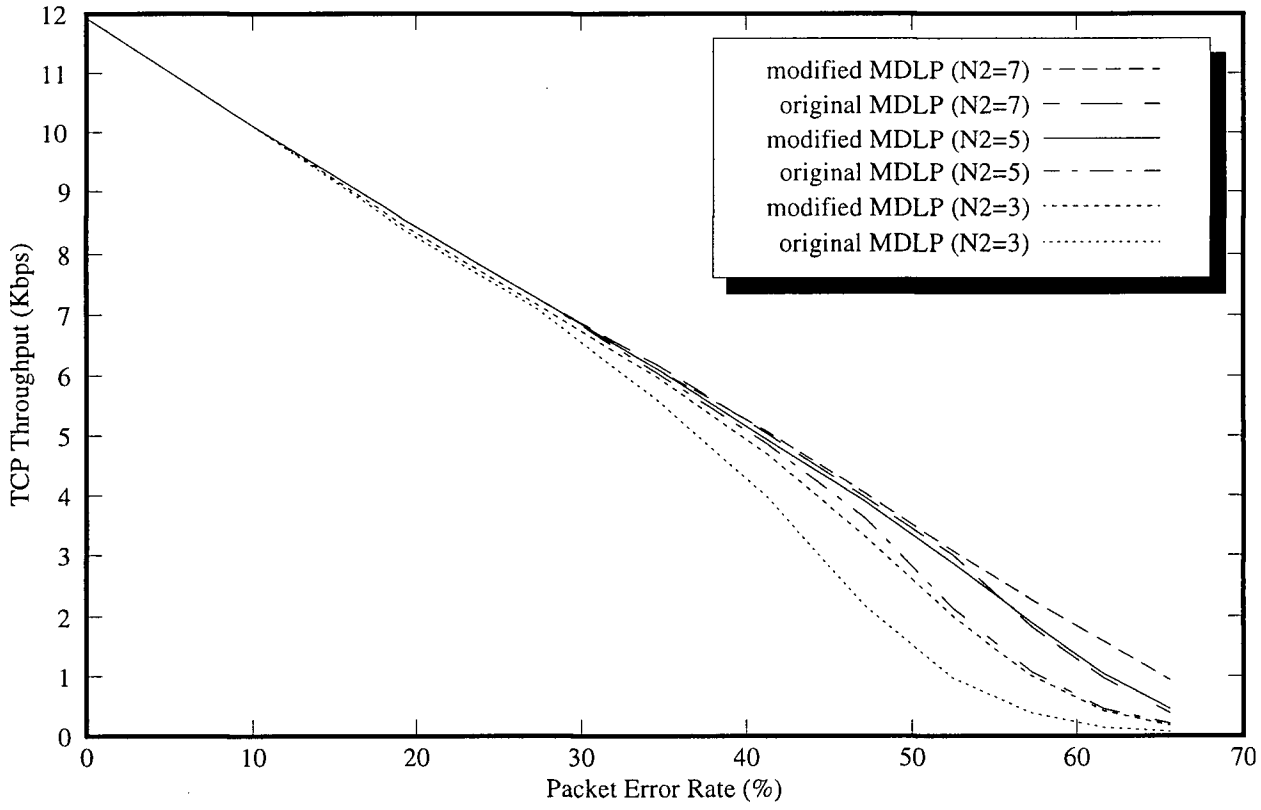

Figure 5.2 Throughput of TCP using the original and modified MDLP under various values of N2

In this section, simulation results of the average transmission delay in the data link layer, the average values of RTT samples and RTO estimates in TCP under different settings of the maximum number of retransmissions and maximum recovery delay are discussed. Performance evaluation on TCP throughput using non-sequencing MDLP and RLP is provided.

Figure 5.3 shows the effect of the maximum recovery delay (*mrd*) in RLP on the average transmission delay of the link layer. In general, the average transmission delay of the link layer increases with the time taken by the link layer for error recovery. When PER is small, most of the lost packets can be recovered in the link layer within 2.0s and therefore the average delay remains about the same if the *mrd* is increased from 2.0s to 5.0s. When PER is high, a larger *mrd* allows the link layer to have longer time to recover a lost packet, resulting in higher average transmission delay.

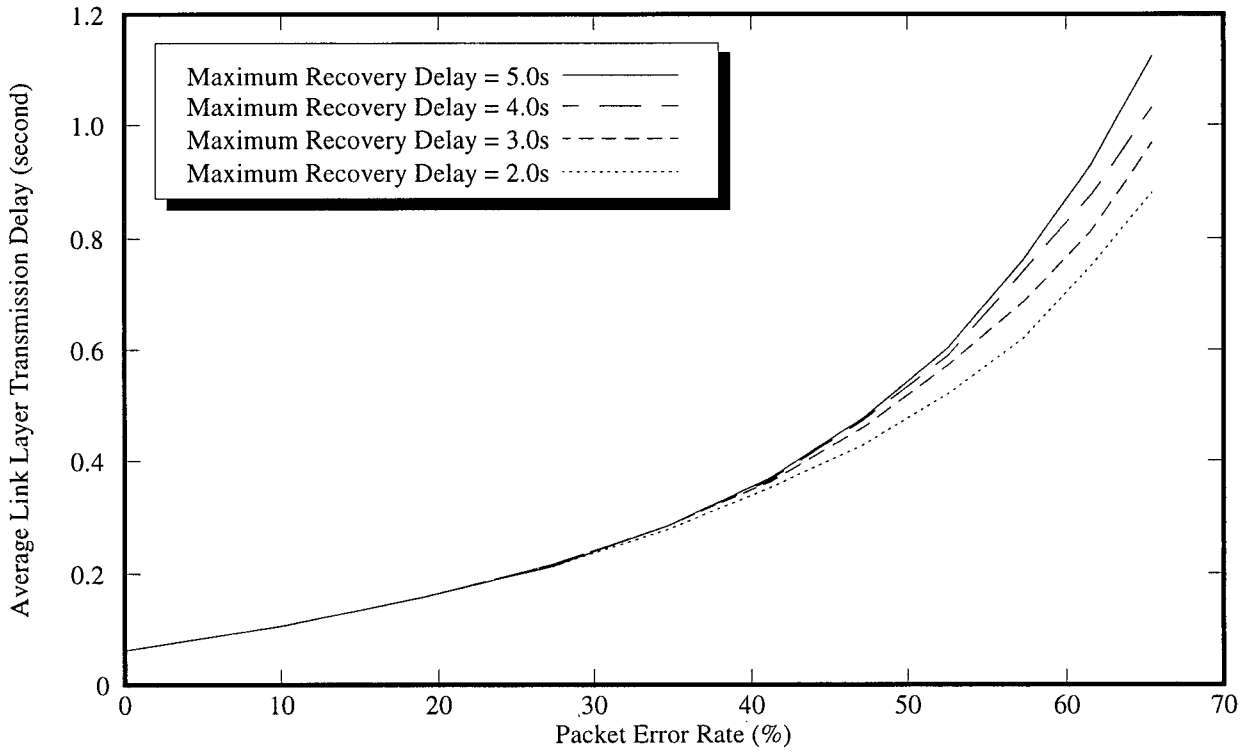

Figure 5.3 Average transmission delay of link layer under various values of maximum recovery delay versus PER

The effects of the maximum recovery delay (*mrd*) in RLP on the average value of RTT sample in TCP is shown in Figure 5.4. In general, the average RTT value increases with the time taken by the link layer for error recovery (as explained in 4.2). With *mrd* = 2.0s, the average RTT increases with PER until PER reaches about 40%. Further increase in PER does not raise the average RTT because the link layer gives up retransmission of lost packets which have already taken more than 2.0s to recover. With *mrd* = 5.0s, the average RTT increases with PER through a wider range of PER, from 0 to 60%, because the link layer is given more time to recover losses.

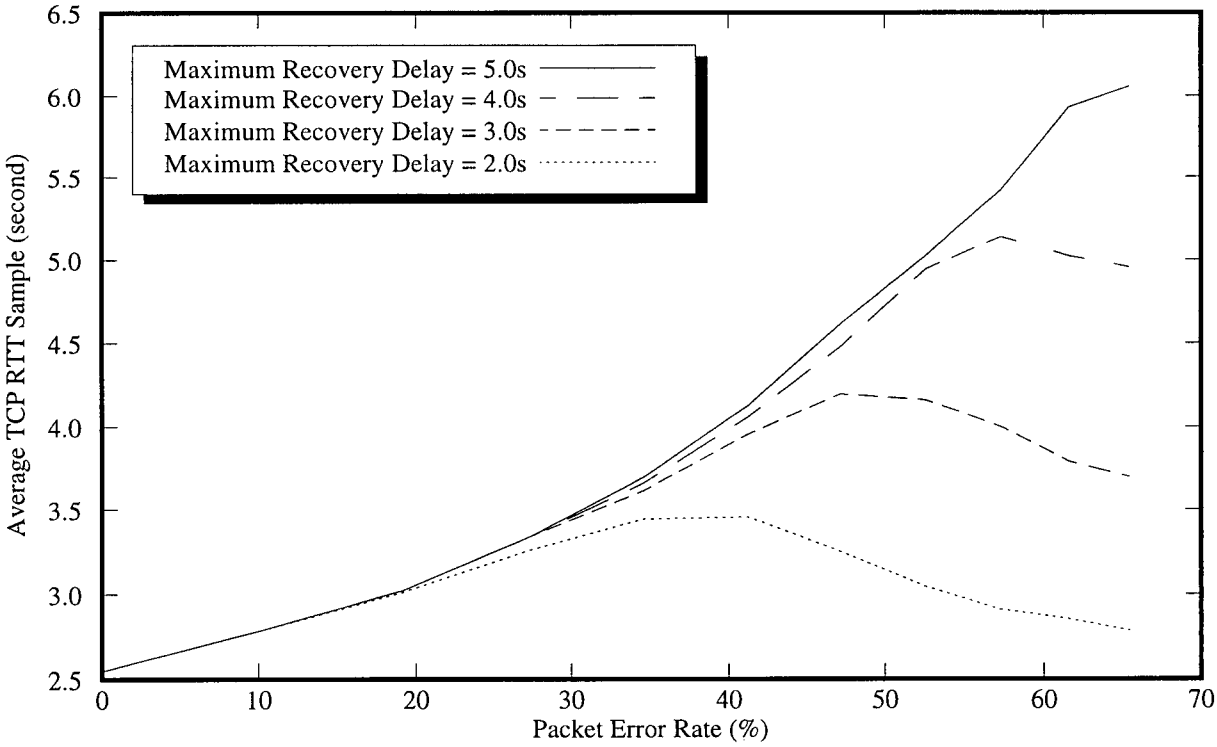

Figure 5.4 Average RTT in TCP under various values of maximum recovery delay versus PER (RLP)

Figure 5.5 shows that the average RTT in TCP with best-effort MDLP exhibits the same behavior as that with RLP. With N2 = 2, the average RTT increases with PER to a maximum value of about 3.0s. With larger N2, the average RTT increases with PER through a wider range of PER and reaches higher values.

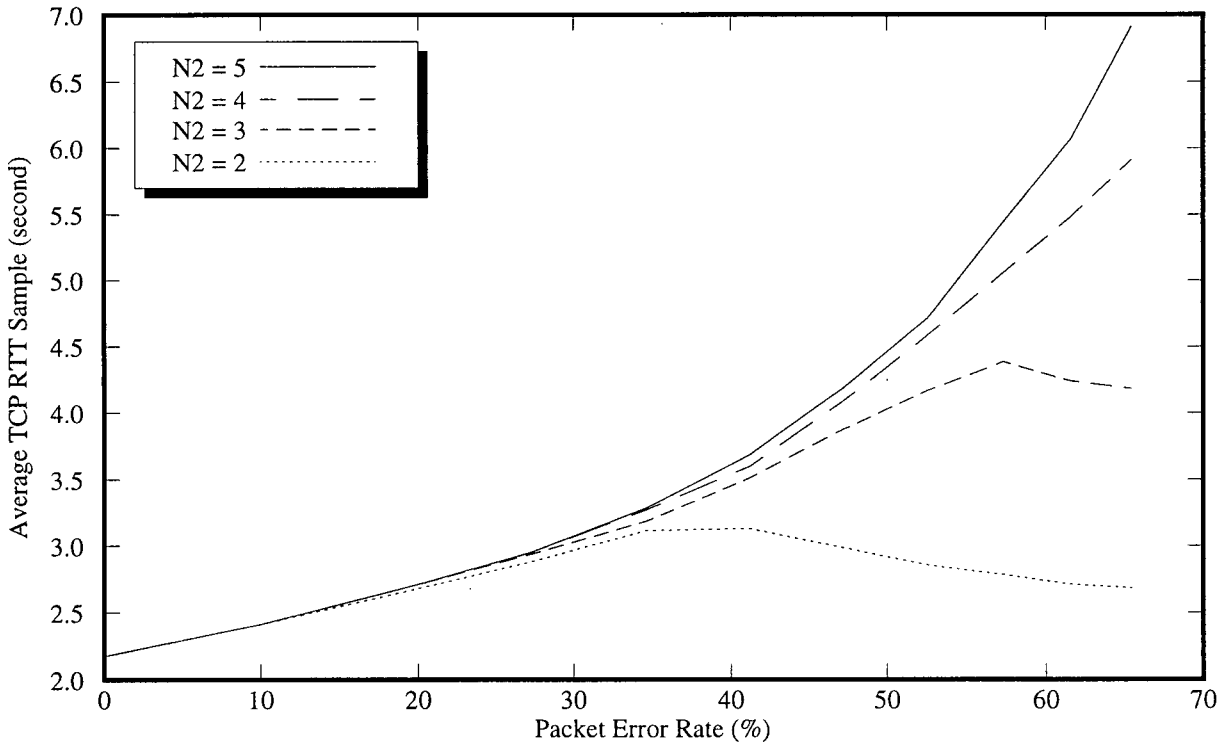

Figure 5.5 Average RTT in TCP under various values of N2 versus PER (MDLP)

Figure 5.6 and Figure 5.7 show that the average RTO in TCP exhibits similar behavior as the average RTT. When maximum recovery delay = 2.0s or N2 = 2, average RTO reaches some maximum value with large PER. However, when maximum recovery delay or maximum number of retransmissions is large, the average RTO keeps on increasing with PER over a wide range of PER.

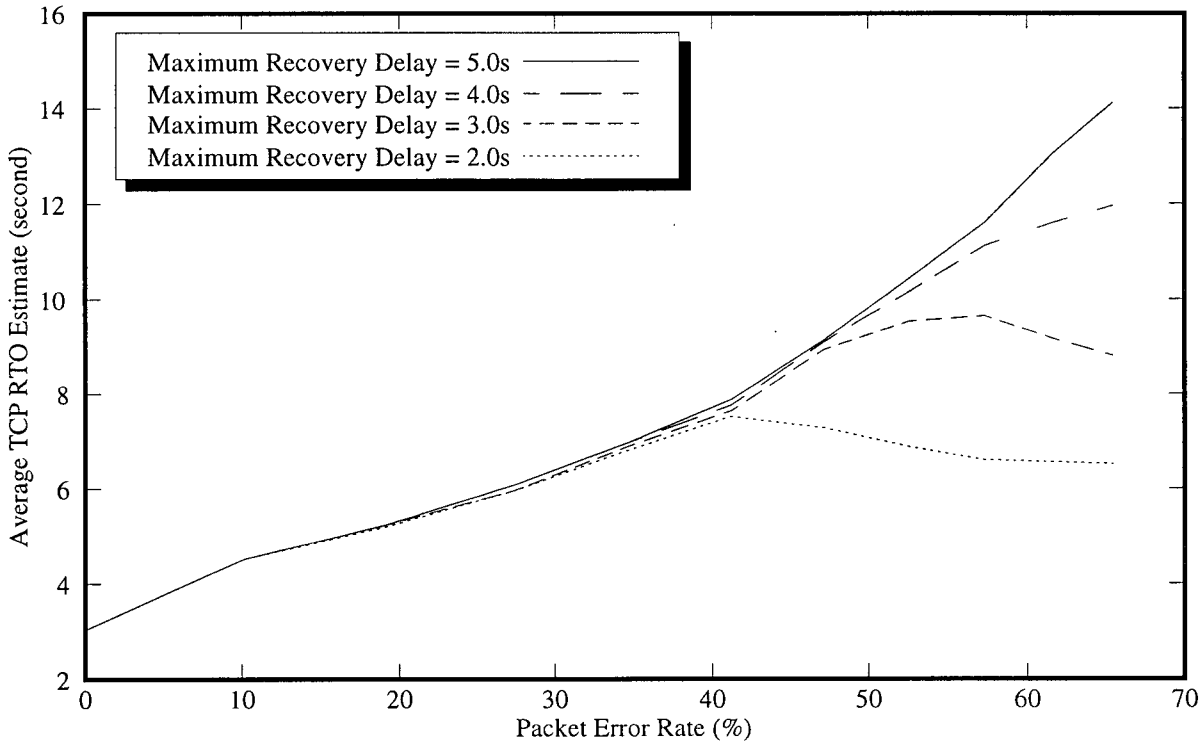

Figure 5.6 Average RTO in TCP under various values of maximum recovery delay versus PER (RLP)

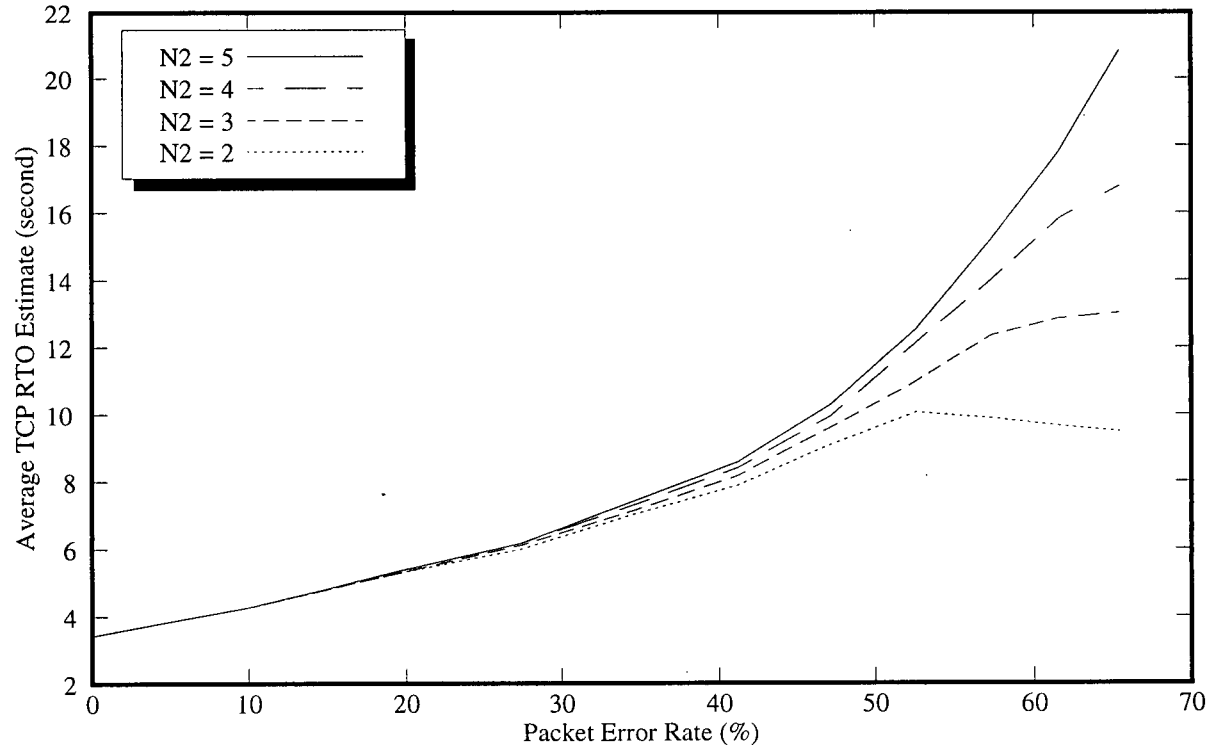

Figure 5.7 Average RTO in TCP under various values of N2 versus PER (MDLP)

Previous discussions on the average transmission delay of link layer, and the RTT and RTO of TCP indicate that while decreasing the maximum recovery delay or the maximum number of retransmissions reduces the delay of both link layer and TCP, it reduces the reliability of the link layer as seen by TCP, causing TCP to recover more losses. Since TCP considers each loss as an indication of network congestion and hence slows down the traffic, TCP may suffer from reduced throughput performance. On the other hand, a link layer with higher reliability hides from TCP most of the wireless losses, but results in higher RTO values in TCP. With a higher RTO, TCP takes longer to recover losses by time-out retransmissions. Since losses affecting TCP do not only come from the wireless link but also from network congestions, a higher RTO value can slow down the recovery of congestion losses in TCP by time-out retransmission.

This is illustrated in Figure 5.8 for best-effort RLP, which shows that when PER is high, TCP does not necessarily have a higher throughput if the maximum recovery delay (*mrd*) is larger. With *mrd* = 2.0s, TCP needs to recover more wireless losses that the data link layer has given up retransmissions, giving lower throughput when PER is high. With *mrd* = 4.0s and 5.0s, a higher RTO value at high PER (see Figure 5.6) makes TCP recover congestion losses relatively slowly, giving lower throughput than with *mrd* = 3.0s. The best choice for *mrd* is about 3.0s.

Figure 5.9 show that the TCP throughput with best-effort MDLP exhibits the same behavior as that with RLP. With N2 = 2 and 3, TCP needs to recover more wireless losses that the data link layer has given up retransmissions, giving low throughput when PER is high. With N2 = 5, a higher RTO value at high PER (see Figure 5.7) makes TCP recover congestion losses relatively slowly, giving lower throughput than with N2 = 4. The best choice for N2 is 4.

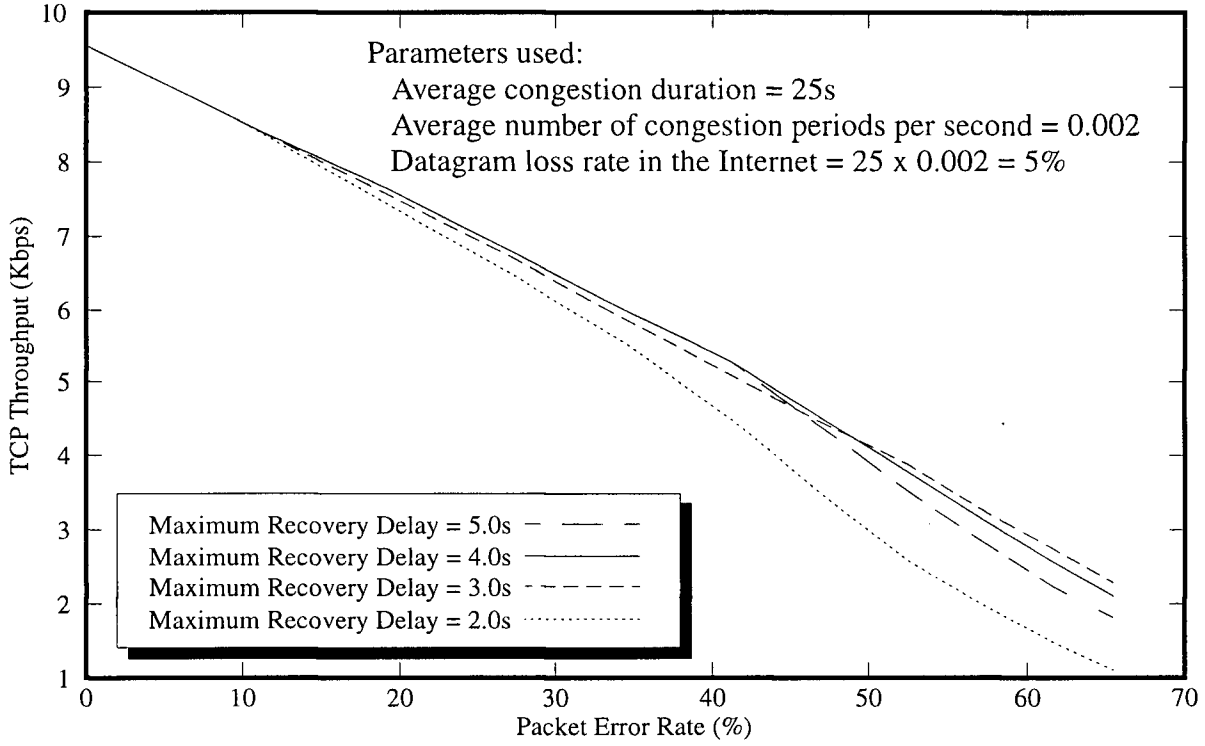

Figure 5.8 Throughput of TCP under various values of maximum recovery delay versus PER (RLP)

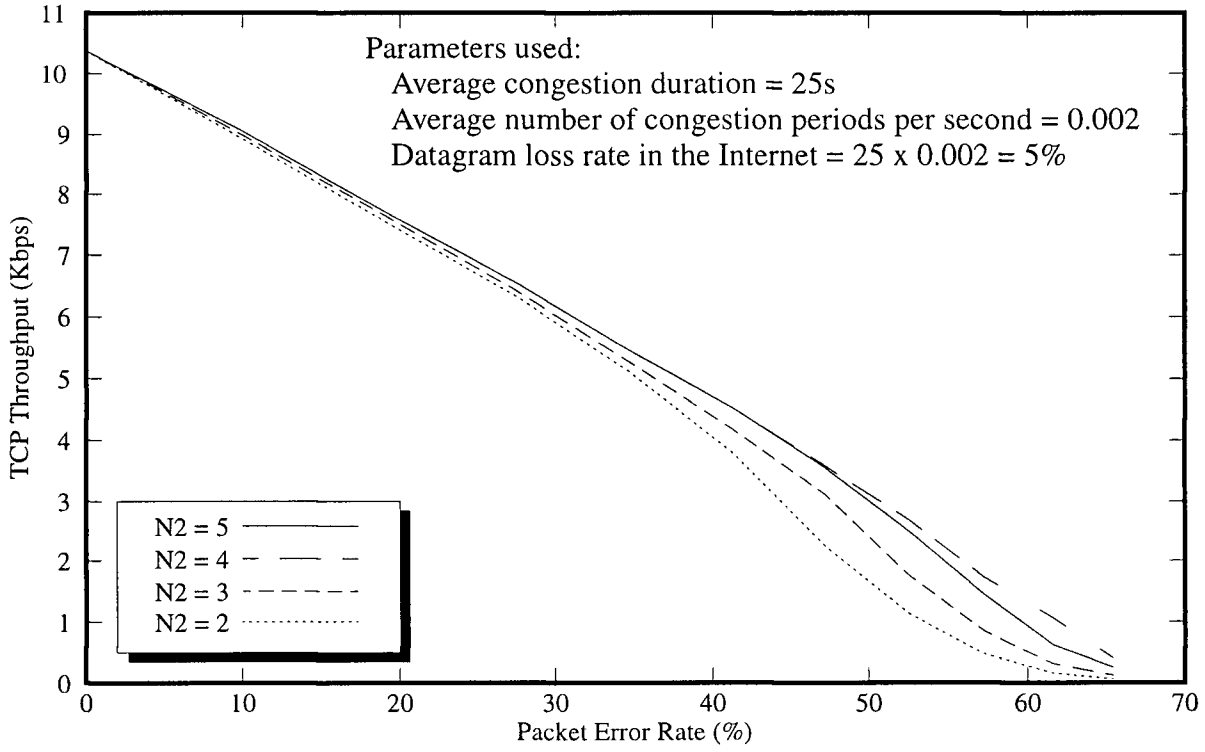

Figure 5.9 Throughput of TCP under various values of N2 versus PER (MDLP)

Figure 5.10 indicates that the TCP throughput with a 2% datagram loss rate in the Internet, as expected, is larger than the TCP throughput with a 5% Internet loss rate in Figure 5.8. In general, a smaller *Internet loss rate* reduces the number of time-out retransmissions for congestion losses and thus lessens the adverse effect of the increased RTO with larger mrd. The best choice for mrd in Figure 5.10 is 4.0s, which is larger than the 3.0s in Figure 5.8.

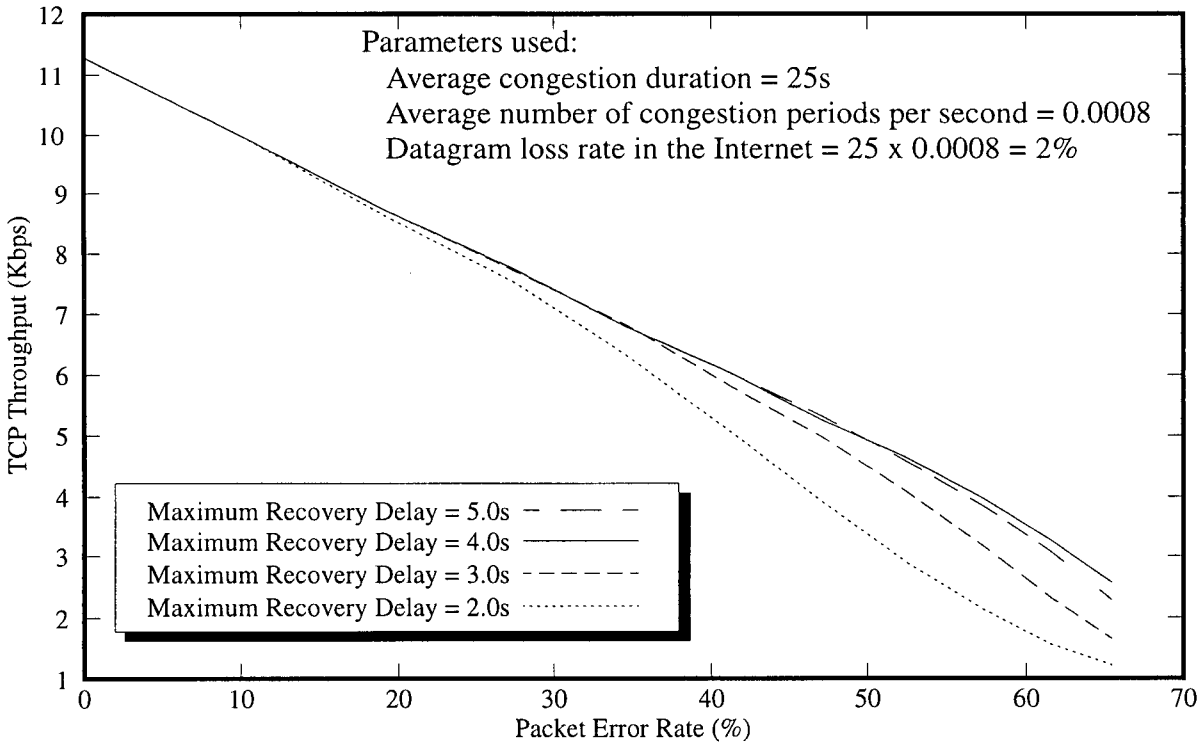

Figure 5.10 Throughput of TCP under various values of maximum recovery delay with 2% Internet loss rate

Figure 5.11 shows the TCP throughput with the same Internet loss rate as that in Figure 5.8 but a smaller *average congestion duration*. In general, a smaller congestion duration reduces the number of *consecutive* time-out retransmissions in TCP and thus lessens the adverse effect of the increased RTO with larger mrd. The best choice for mrd in Figure 5.11 is 4.0s, which is larger than the 3.0s given by Figure 5.8.

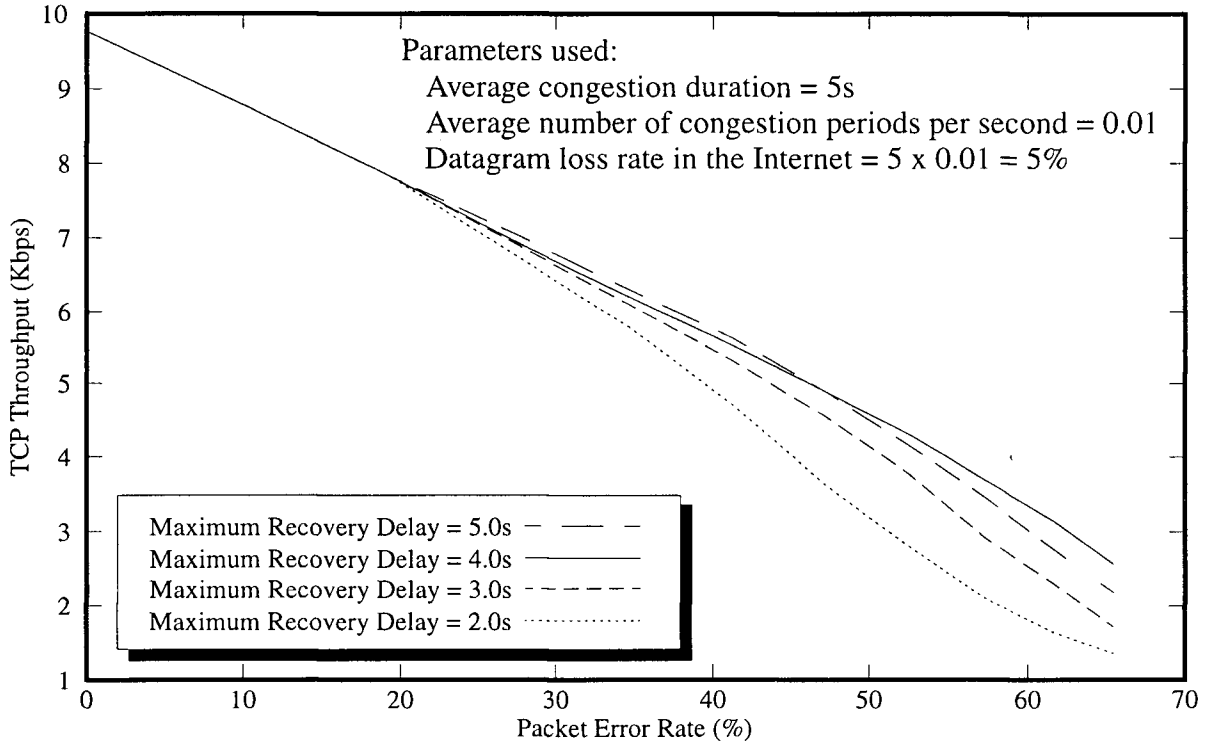

Figure 5.11 Throughput of TCP under various values of maximum recovery delay with 5s average congestion duration

5.3 Summary of simulation results using best-effort data link protocols

In summary, data link protocols using best-effort retransmissions recover as many lost packets over the wireless links as possible. Losses that cannot be recovered by the link layer will need to be eventually recovered in the TCP layer. Receiver-initiated data link protocols use the maximum recovery delay (maximum time that sender is allowed to retransmit a frame) to control the reliability of link layer provided to TCP. Data link protocols in which retransmissions are initiated by the sender or both hosts use the maximum number of retransmissions to control the reliability.

Lower reliability in the link layer causes TCP to experience more wireless losses, resulting in degradation in TCP throughput. However, higher reliability hides from TCP most of the

wireless losses, but gives higher RTO estimates in TCP. With higher RTO estimates, TCP takes longer time to recover losses by time-out retransmissions. Since losses affecting TCP do not only come from the wireless link but also from network congestion, higher RTO estimates can slow down the recovery of congestion losses by TCP. In conclusion, data link layer should put a limit on wireless-loss recovery by employing best-effort retransmission. With a suitable division of the wireless-loss recovery function between TCP and link layer, TCP performance can be maximized.

Chapter 6 Conclusion

TCP does not perform well in networks with high packet error rates, such as those with wireless links, because TCP incorrectly interrupts wireless losses as network congestion losses. Wireless losses make TCP unnecessarily initiate congestion control mechanism which slows down the TCP traffic and cause frequent time-out retransmissions, resulting in poor performance in the form of low throughput and high interactive delay.

Link layer schemes, which employ data link protocols in the base stations and mobile hosts to retransmit lost packets over the wireless link, may be employed to hide wireless losses from TCP as much as possible. However, the problem of competing retransmissions between TCP and data link layer may occur, causing unnecessary duplications in TCP retransmission and significant degradation in TCP performance. A detailed analysis of using link layer retransmissions to improve the end-to-end performance of TCP in wireless networks has been presented in this thesis. Simulation results show that employing existing data link protocols like CDPD MDLP and RLP for local retransmissions over wireless link can help to improve the throughput and end-to-end delay performance of TCP. The conclusion is that the problem of competitive retransmissions between TCP Reno and link layer is not obvious for slow speed data links using the more effective selective-reject ARQ for link layer retransmissions.

It is also discovered that non-sequencing data link protocols do not necessarily degrade TCP performance. Instead, re-sequencing buffers and complex logic for handling out-of-sequence packets that would otherwise be needed for a sequencing link layer protocol can be avoided, thus simplifying protocol implementation.

Three different retransmission schemes, LAPB, MDLP and RLP, are used to evaluate the TCP throughput performance with link layer recovery. It is found that, the receiver-initiated retransmission scheme of RLP gives better TCP performance in an error-prone wireless environment because the receiver uses periodic status messages to inform the sender exactly which frame is received and which frame is missing, reducing unnecessary duplications from retransmissions.

Finally, it is shown that frequent link layer resets can degrade the throughput performance of TCP. To overcome this problem, modifications to both sender-initiated and receiver-initiated link layer protocols are proposed for best-effort retransmissions. The effects of link layer reliability on TCP throughput and RTT estimations are investigated. The conclusion is that low reliability causes TCP to recover more losses but high reliability may slow down TCP time-out retransmissions for recovery of network congestion losses. With a suitable division of the wireless-loss recovery function between TCP and link layer, the TCP throughput can be maximized and the adverse effect of link layer recovery can be minimized.

Possible future works:

The following areas of TCP with link layer recovery are suggested for further investigations:

- Performance analysis of TCP with link layer recovery employing both retransmissions and FEC (e.g. the AIRMAIL protocol in [26]), and
- performance analysis taking into considerations of TCP/IP header compression for IP fragments transmitted over wireless link [32].

Bibliography

[1] W. Stallings: *Data and Computer Communications*, 4th edition. Macmillan, New York, 1994.

[2] T. Socolofsky and C. Kale, "TCP/IP tutorial", IETF RFC 1180, Jan. 1991.

[3] J. B. Postel, "Internet Protocol", IETF RFC 791, Sep. 1981.

[4] J. B. Postel, "Internet Control Message Protocol", IETF RFC 792, Sep. 1981.

[5] J. B. Postel, "User Datagram Protocol", IETF RFC 768, Aug. 1980.

[6] J. B. Postel, "Transmission Control Protocol", IETF RFC 793, Sep. 1981.

[7] W. R. Stevens, "TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms", IETF RFC 2001, Jan. 1997.

[8] W. R. Stevens: *TCP/IP Illustrated, Vol. 1: The Protocols*. Addison-Wesley, 1994.

[9] G. R. Wright and W. R. Stevens: *TCP/IP Illustrated, Vol. 2: The Implementation*. Addison-Wesley, 1995.

[10] P. Manzoni, D. Ghosal and G. Serazzi, "Impact of Mobility on TCP/IP: An Integrated Performance Study", *IEEE JSAC*, Vol. 13, No. 5, Jun. 1995.

[11] A. DeSimone, M. C. Chuah and O. C. Yue, "Throughput Performance of Transport-Layer Protocols over Wireless LANs", *Proc. IEEE GLOBECOM*, pp. 542-549, Nov. 1993.

[12] MIL 3 Inc., *Optimized Network Engineering Tools (OPNET), Release 3.0.B*, 1997.

[13] "Mobile Data Link Protocol", Part 403, CDPD System Specification, Release 1.1, Jan. 19, 1995.

[14] V. Jacobson, "Congestion Avoidance and Control", *Proc. ACM SIGCOMM*, Aug. 1988.

[15] P. Karn and C. Partridge, "Improving Round-Trip Time Estimates in Reliable Transport Protocols", *Proc. ACM SIGCOMM*, Aug. 1987.

[16] P. Karn, "The Qualcomm CDMA Digital Cellular System", *Proc. 1993 USENIX Symp. on Mobile and Location-Independent Computing*, pp. 35-40, Aug. 1993.

[17] N. Samaraweera and G. Fairhurst, "Explicit Loss Indication and Accurate RTO Estimation for TCP Error Recovery using Satellite Links", *IEE Proc. Communications*, Vol. 144, No. 1, pp. 47-53, Feb. 1997.

[18] A. Bakre and B. R. Badrinath, "I-TCP: Indirect TCP for Mobile Hosts", *Proc. IEEE ICDCD*, May 1995.

[19] H. Balakrishman, S. Seshan, E. Amir and R. H. Katz, "Improving TCP/IP Performance over Wireless Networks", *Proc. ACM MOBICOM*, Nov. 1995.

[20] R. Careres and L. Iftode, "The Effects of Mobility on Reliable Transport Protocols", MITL Techinal Report, MITL-Tr-73-93, Nov. 1993.

[21] C. E. Perkins and P. Bhagwat, "A Mobile Networking System based on Internet Protocol", *IEEE Personal Communications*, First Quarter, 1994.

[22] K. Pahlavan and A. H. Levesque: *Wireless Information Networks*. John Wiley & Sons, 1995.

[23] "Medium Access Protocol", Part 402, CDPD System Specification, Release 1.1, Jan. 19, 1995.

[24] F. A. Tobgi, "Distributions of Packet Delay and Interdeparture Time in Slotted Aloha and Carrier Sense Multiple Access", *ACM Journal*, Vol. 29, No. 4, pp. 907-927, Oct. 1982.

[25] S. L. Beuerman and E. J. Coyle, "The Delay Characteristics of CSMA/CD Networks", *IEEE Trans. Communications*, Vol. 36, No. 5, May 1988.

[26] E. Ayanoglu, S. Paul, T. F. LaPorta, K. K. Sabnani and R. D. Gitlin, "AIRMAIL: A link-layer protocol for wireless networks", *ACM Wireless Networks*, No. 1, pp. 47-60, Feb. 1995.

[27] S. Paul, E. Ayanoglu, T. F. LaPorta, K. W. H. Chen, K. K. Sabnani and R. D. Gitlin, "An Asymmetric Protocol for Digital Cellular Communications", *Proc. IEEE INFOCOM*, pp. 1053-1062, 1995.

[28] R. J. Deasington: *X.25 Explained - protocols for packet switching networks*, 2nd edition. John Wiley & Sons, 1986.

[29] B. D. Fritchman, "A binary channel characterization using partitioned Markov Chains", *IEEE Trans. Information Theory*, pp. 221-227, 1967.

[30] D. D. Clark, "IP Datagram Reassembly Algorithms", IETF RFC 815, Jul. 1982.

[31] H. Balakrishman, V. N. Padmanabhan, S. Seshan and R. H. Katz, "A Comparison of Mechanisms for Improving TCP Performance over Wireless Links", *Proc. ACM SIGCOMM*, Aug. 1996.

[32] V. Jacobson, "Compressing TCP/IP Headers for Low-Speed Serial Links", IETF RFC 1144, Sep. 1990.

Appendix A List of Abbreviations

ACK	Acknowledgment
ARQ	Automatic Repeat Request
BER	Bit Error Rate
CDMA	Coded Division Multiple Access
CDPD	Cellular Digital Packet Data
DLC	Data Link Control
FDMA	Frequency Division Multiple Access
FEC	Forward Error Correction
ICMP	Internet Control Message Protocol
IP	Internet Protocol
LAN	Local Area Network
LAPB	Link Access Protocol - Balanced (in X.25 networks)
MAC	Medium Access Control
MDLP	Mobile Data Link Protocol (in CDPD)
MSC	Mobile Switching Center
MSS	Maximum Segment Size (in TCP)
MTU	Maximum Transfer Unit
N2	Maximum Number of Retransmissions (in MDLP)
OSI	Open Systems Interconnection (Reference Model)
PER	Packet Error Rate
RLP	Radio Link Protocol (in [27])
RTO	Retransmission Time-out
RTT	Round Trip Time
SRTT	Smoothed Round Trip Time
TCP	Transmission Control Protocol
TDMA	Time Division Multiple Access
UDP	User Datagram Protocol

Appendix B OPNET Simulation Models

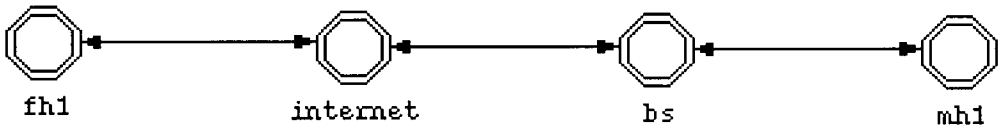

Figure B.1 The network model of a TCP connection between a fixed host and a mobile host

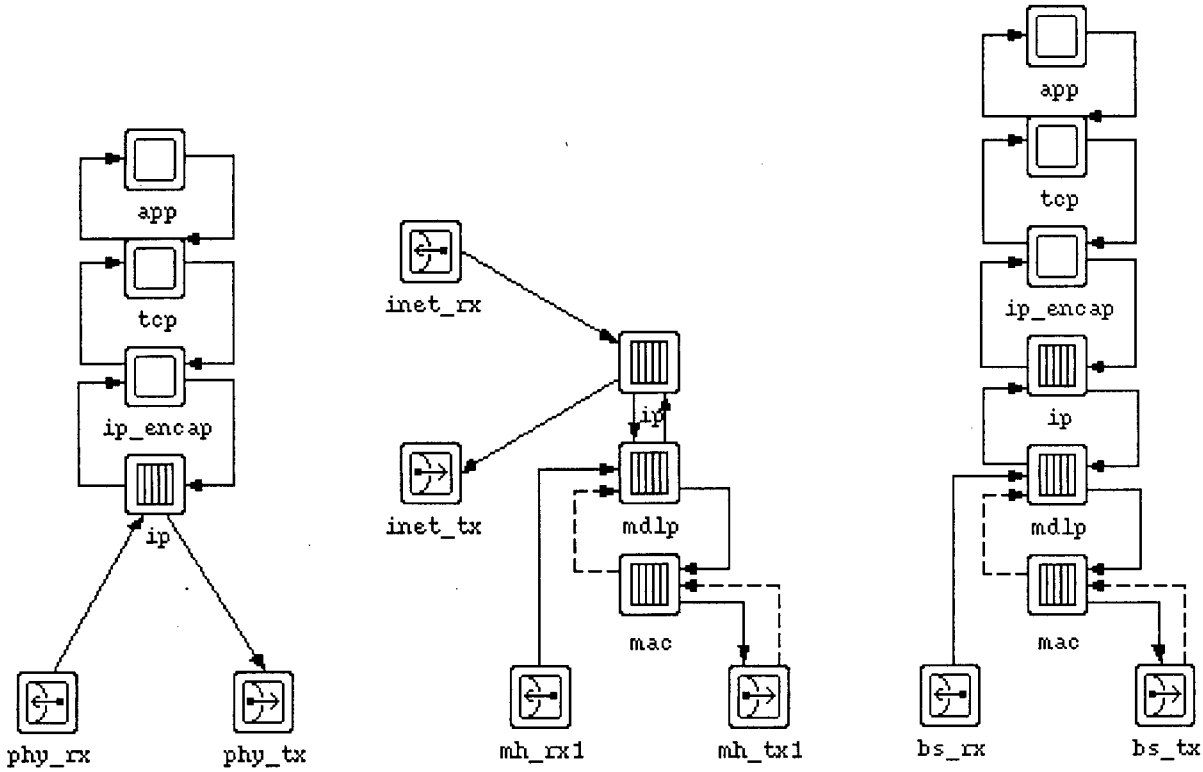

Figure B.2 The node models of the fixed host (left), base station (middle) and mobile host (right)

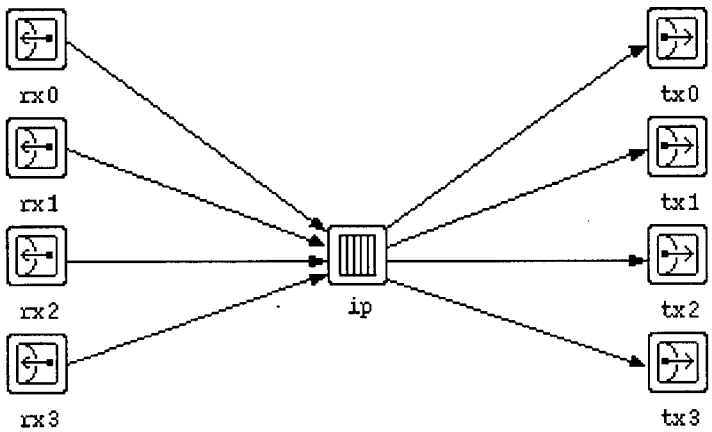

Figure B.3 The node model of an Internet router

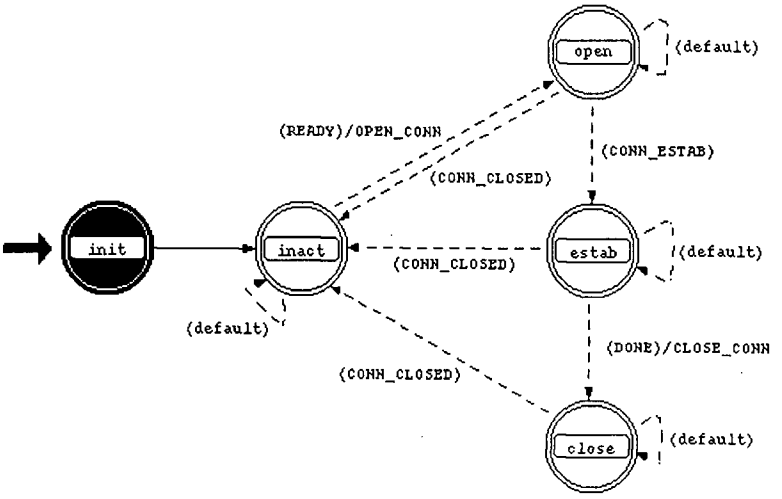

Figure B.4 The process model of a TCP application

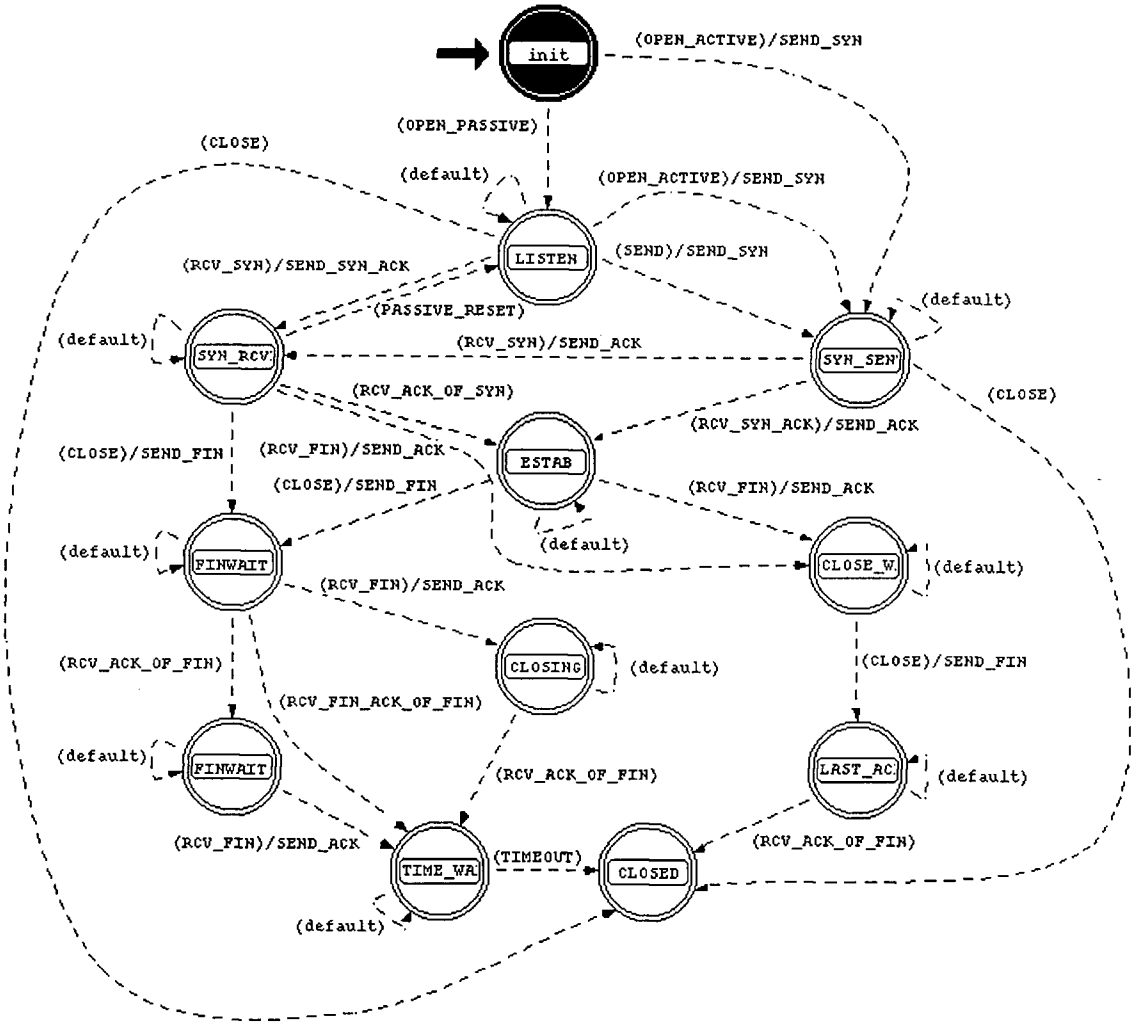

Figure B.5 The process model of TCP

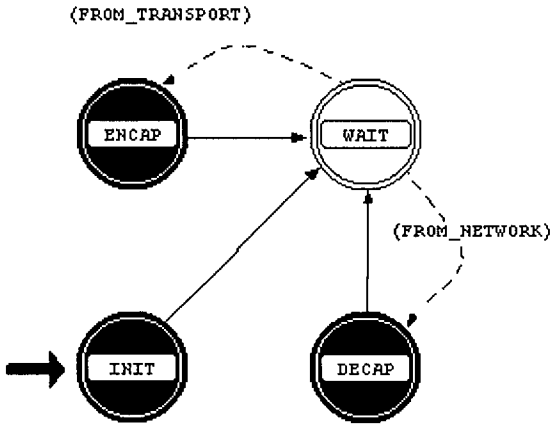

Figure B.6 The process model of IP header encapsulation / decapsulation

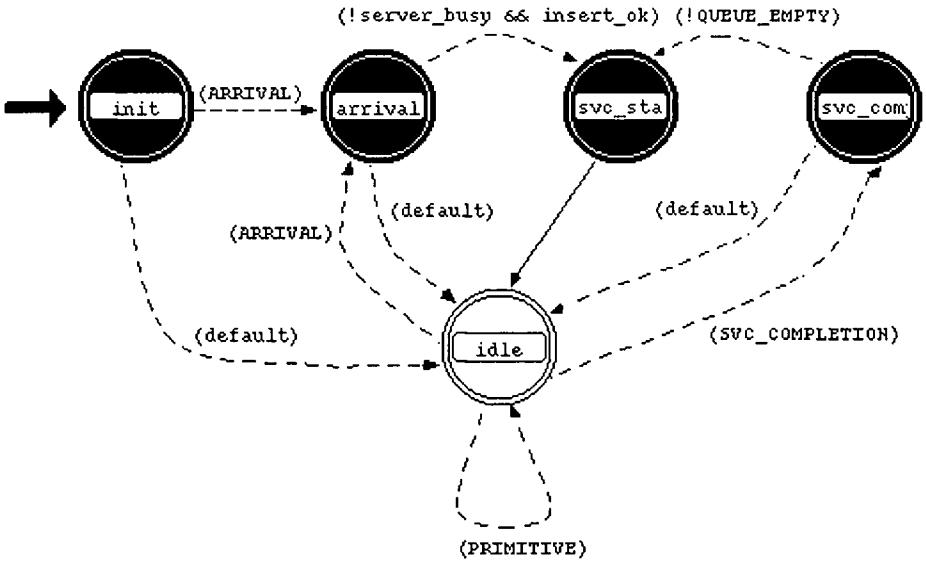

Figure B.7 The process model of IP routing and fragmentation

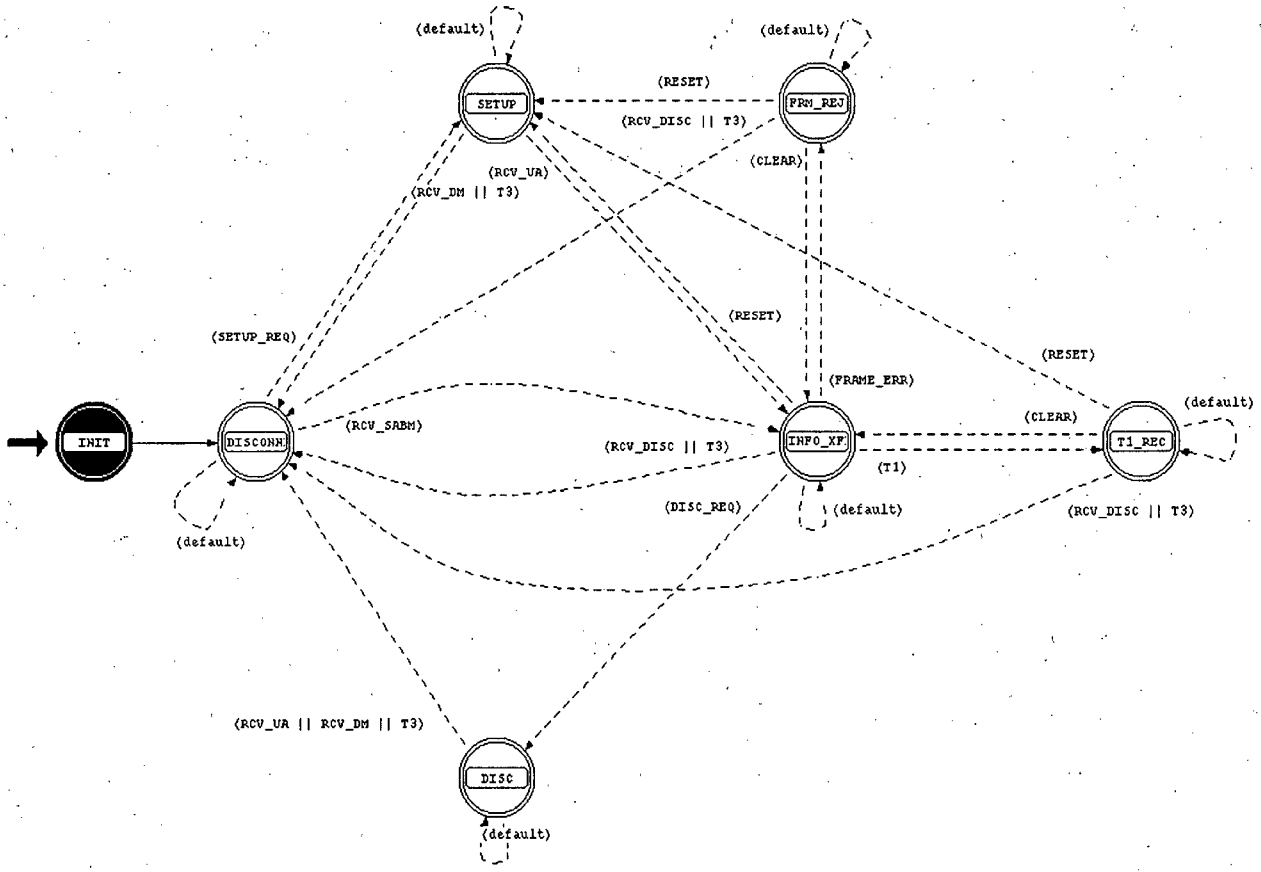

Figure B.8 The process model of CDPD MDLP

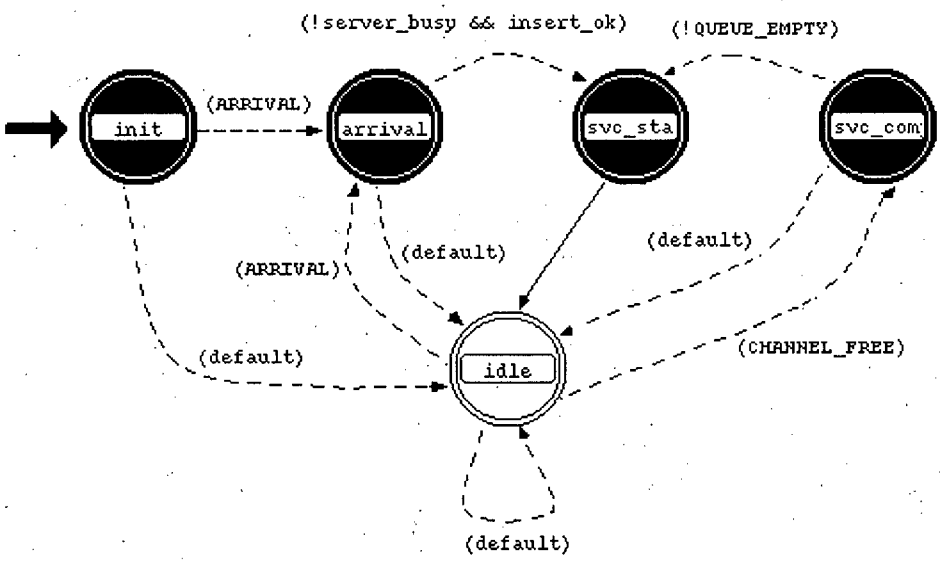

Figure B.9 The process model of MAC