MODELING AND ANALYSIS OF TCP IN SATELLITE AND MOBILE DATA NETWORKS INTERWORKING WITH THE INTERNET

by

EVA Y. F. LEUNG

B.A.Sc.(E.E.), University of Maryland, College Park, U.S.A., 1993

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

In presenting this thesis in partial fulfillment 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 April 27, 1998

Abstract

The Transmission Control Protocol (TCP) is the most widely used transport protocol for the Internet because of its good performance over wireline networks. However, TCP performance is seriously degraded in many interconnected heterogeneous wireless and wired networks because their large delays and high packet loss rates violate many of the original TCP assumptions.

This thesis presents an analysis and a comparison of the performance of different TCP implementations in networks with satellite links and mobile links interconnected with remote LANs over the Internet, studies the effects of the critical network elements and the chosen TCP parameters on the TCP throughput performance, and obtains the optimal values for the chosen TCP parameters for the different TCP implementations. Compared with TCP Reno and TCP RFC, TCP Vegas not only has a better congestion avoidance mechanism to prevent network congestion and significantly reduce the probability of creating its own losses, but it also has a faster error recovery mechanism to quickly retransmit lost packets.

Simulation results show that the satellite network elements and the TCP parameters have greater effect on the throughput of TCP Vegas than on the other implementations under most conditions for networks with satellite links. However, for networks with mobile links, the throughput performance of both TCP Vegas and TCP Reno is strongly affected by the mobile network elements and the TCP parameters. After jointly optimizing the key TCP parameters, the simulation results show that TCP Vegas has more than 100% better throughput than the other TCP implementations for networks with satellite links under a condition of high packet loss rates and long end-to-end delays. However, it just has an approximately 5% better throughput than TCP Reno for networks with mobile links under a condition of high packet loss rates and high mobility rates.

Table of Contents

List of Tables

List of Figures

Acknowledgment

This thesis is dedicated to my parents, Fong Leung and Mee Wah Chu, and my brothers and sisters for continuous support and encouragement for me to pursue my goal. In particular, I would like to express my sincere gratitude to my research supervisor, Dr. Victor C. M. Leung, for his guidance and constant patience throughout my graduate studies at the University of British Columbia. Lastly, I would offer my appreciation to my close friends and my colleagues for their continuous support.

This work was funded by the Canadian Institute of Telecommunications Research (CITR) and Motorola, Inc. through a Research Assistantship provided by Dr. Victor Leung. I would also like to thank Mil 3, Inc. for providing the state-of-art simulation software, OPNET, and technical support throughout this work.

Chapter 1 Introduction

Over the last decade Internet communication has been growing rapidly, providing high-speed network connectivity to a large number of heterogeneous computer applications. However, the existing terrestrial wide area network is wired, and therefore limited both in its wide area coverage and by its inability to link with mobile hosts. To transcend these limitations, wireless communications technologies have been developed for interconnecting stationary or mobile remote hosts. Broadband satellite communications can provide wide area coverage, typically to stationary hosts. Land mobile cellular communications operate over smaller areas, but can link stationary (e.g. Internet) to mobile hosts. Analyses of Internet applications [1][27] indicate that the packet length, terminal activity, and data rate of user terminals may keep changing by many orders of magnitude, thus causing the Internet traffic to change dynamically over a wide range. Large file transfer, multimedia communication, and image conferencing are expected to be the major future network traffic. These applications will require short response time, high throughput performance, and very flexible interconnection architectures.

Therefore, reliable transport protocol performance and flow control analyses of communications end-to-end across both wireless and wired networks are critically needed to ensure the desired quality of service in seamless end-to-end delivery of the data traffic. Because Transmission Control Protocol (TCP) works very well in traditional wireline networks made up of wired links and stationary hosts, it has been considered for use on heterogeneous wireless and wired (HWW) networking environments.

1.1 Background

TCP is currently the most widely used transport protocol for a large spectra of topologies

in wide area networks or in the Internet. TCP [20][28] was originally designed for the slow-speed, unreliable, datagram-based ARPANET where packets were often lost, duplicated or delivered out of sequence. Because computing and networking technologies have been growing and the characteristics of the Internet traffic have been significantly changing, many implementations and options for TCP have been introduced to enhance the service of the Internet Protocol (IP). The different TCP implementations are presented in detail in chapter 2.

TCP employs a window scheme to control packet flow from a sender to a receiver. It also uses a positive acknowledgment with a retransmission scheme for error recovery. By adapting to end-to-end delay and packet loss rate, the error recovery and congestion control mechanisms in TCP have been performing very well in low bit error rate wireline networks such as the Internet. In these networks packet losses are mainly due to network congestion. However, communications over satellite or mobile radio networks are quite different from those over traditional wireline networks [16]. These wireless links do not have the high bandwidth of wired links, have higher latencies, and have much higher error rates. Their packet losses, in contrast to the wired networks, are mainly caused by outages due to connection interruptions. In broadband satellite channels, large data segments can be lost when the channel changes suddenly due to a rapid change in the weather conditions. In the mobile environment, the packet losses are mainly caused by wireless links which are temporarily aborted due to deep fades or due to hard handoffs when the mobile crosses a cell boundary.

Because of these unexpected interruptions, the error recovery and flow control mechanisms in TCP cannot work well. TCP always misinterprets an unexpected increase in delay as packet loss due to network congestion [2][3][4][5][6][7][10]. As a result, TCP first drops the transmission window size to reduce the amount of data in transit through the network. Second, it activates its slow-start algorithm to control the traffic rate at which the traffic grows back to its

previous level. Third, it resets its retransmission timer to a back off interval that doubles with each consecutive time-out.

If a packet is dropped due to an overloading of limited resources in the network, congestion control has to be applied immediately in order to restrict the traffic flow and give the network a chance to clear up. However, if a connection loses a packet due to the unreliable wireless links, which means the packet was corrupted by errors or by a temporary communication pause due to a handoff process, no congestion control is necessary. The sender should not reduce the data flow rate by performing the slow-start mechanism to drastically decrease the congestion window size, thus wasting the available bandwidth and further degrading the system performance. The problem is worsened in severely fading channels [3]. The slow start mechanism will be highly active for a long period of time, thus increasing the frequency of end-to-end communications pauses. Because of the exponential back-off policy, the communication pauses at the sender will be unnecessarily long, especially in high mobility environments [7][11]. Although the slow-start mechanism works well for avoiding network congestion when it is used over terrestrial wireline network (low packet loss rate and small delay), it is very inefficient in high loss rate and highly interactive delay environments.

The long propagation delay in broadband satellite networks is another major problem degrading TCP throughput. With a long propagation delay between a source and a destination, the frequency of sending packets into the network is low because the sender must wait for a long round-trip-delay to receive an ACK before it can send or retransmit packets. If it is using the slow-start mechanism for congestion prevention, TCP further limits the size of the transmit window for every packet loss detected by time-out. Therefore, the utilization of a long propagation delay satellite channel is degraded by the limited TCP window size in combination with the TCP maximum segment size.

1.2 Motivation

Recently there has been significant increase in activity in the area of wireless communications with both stationary and non-stationary hosts interworked with the Internet. TCP is the most widely used reliable transport protocol and has recently been considered for supporting hosts in heterogeneous networks. However, TCP performance is seriously degraded in wireless communications environments because networks with wireless links and mobile hosts seriously violate many of the original TCP assumptions.

In order to ensure the desired quality of service for the data traffic and to exploit the inherent strengths of the heterogeneous networks, it is necessary to perform a detailed performance analysis of TCP in the heterogeneous wireless and wired (HWW) data communications networks under various conditions. TCP has gone through several refinements to improve performance in the highly dynamic Internet traffic environments. The core mechanisms such as flow control and error recovery in different TCP implementations have been modified. Therefore, the performance analysis of different TCP implementations for the networks with satellite links and mobile links interconnected with remote LANs over the Internet is needed. Because a lot of mutually dependent TCP parameters strongly affect the TCP performance, optimization of TCP parameters is also needed for different TCP implementations to yield better throughput performance.

1.3 Previous Work

Many performance analyses have been done for TCP/IP in an interconnection satellite network environment and in a mobile computing environment. A number of studies have been reported in the literature dealing with various aspects required to support hosts in the satellite network environment and mobile computing environments. Several design alternatives have been

proposed to improve the poor end-to-end TCP performance over a network with wireless links.

The authors in [5] presented the performance evaluations in the areas of satellite link delays and bit error rates for the TCP/IP protocol stack and the XTP implementation. In this paper, the TCP implementation based its error recovery strategy on the a Go-back-N protocol; while the XTP [12] implementation based its error recovery strategy on the selective repeat protocol with a sliding window flow control. From the experimental measurement, the authors' conclusions are: 1) The window size has a stronger influence on TCP throughput than on XTP throughput. Large window sizes provide a better channel utilization on high delay links at a relatively low bit error rate. 2) A retransmission ambiguity problem of the TCP implementation makes it more sensitive to a larger delay condition than the XTP implementation. 3) XTP has better performance than TCP due to the use of a selective repeat algorithm of retransmissions for corrupted or lost packets even under a high bit error rate condition. The slow start mechanism of TCP is the main reason for degradation of its performance in HWW networks.

In [3], the authors analyzed the throughput performance of different TCP error control strategies, which are the NAK option, the selective repeat option, and the Shutter XOR selective repeat strategy, while switching off the TCP congestion control scheme for LAN interconnection through a satellite channel. In the simulation the authors found that the throughput was increased by explicit error detection mechanisms because of the reduction of unnecessary retransmissions. The selective repeat option has a higher throughput than the NAK option in the case of high bit error probabilities over a bandwidth-delay dominated network, and the Shutter-XOR selective repeat strategy has slightly better throughput than the selective repeat option at a high bit error rate. At a bit error rate of 5×10^{-6}, the optimal value of the maximum window size for TCP with selective repeat error control strategies is about 64 Kilobytes under the simulation conditions. The throughput efficiency cannot be increased by further increasing this value.

The authors of [2][21] found that the transport protocol are very sensitive to probability of bit error and the altitude of the satellites. Bigger window sizes can improve the end-to-end throughput performance at a relatively low bit error rate. In [4], some key observations were made about the file transfer in a LAN interconnection through a satellite channel. From the simulation results, the authors found that the flow control window in the TCP/IP is the bottleneck of the system if the bridge is implemented for the use of a single pair of workstations. However, the bottleneck will move to the bridge due to its queueing delay, especially at a smaller window size, if the bridge is shared by multiple workstation pairs.

In [7], the effects of mobility on performance of the TCP-Tahoe implementation was studied by measuring TCP behavior in a wireless networking testbed. A TCP connection was initiated between a mobile host connected to a 2-Mbits/sec WaveLAN local area network and a stationary host connected to a 10-Mbits/sec Ethernet local area network. The motion across cell boundaries was simulated with the mobile host switching cells every 8 beaconing periods. The authors found that cell crossings cause increased delays and packet losses. TCP interprets this delay as an indication of network congestion, thus activating the slow-start algorithm to restrict the traffic flow. Because of this misinterpretation, the throughput performance of TCP drops significantly with increased mobile host motion. During handoff pauses, TCP transmits no new data and transport-level communication comes to a halt; as a result, the throughput performance of TCP for a 1-second rendezvous delay is degraded to approximately one-third of the TCP throughput under the no handoff condition. The authors of [7] also investigated the effect of having a fast retransmit approach described in [10] in the TCP implementation, and showed that the mechanism could reduce the pauses in communication from 2.8 to 1.2 seconds for a 1-second rendezvous delay. However, this approach, which is dependent on network layer information, can only deal with handoffs but not with the error characteristics of the wireless links.

In [6], a simulation study of the impact of mobility on performance of the TCP-Tahoe implementation is presented. A queueing network model with the characteristics of Internet delay is developed. From the simulation results, the authors found that the packet losses during handoffs can significantly drop the throughput efficiency especially in highly mobile environments and high bandwidth connections. Large window sizes such as 64Kbyte, further reduce the mean throughput by approximately 10 percent even in a host-limited network. The authors concluded that the loss of throughput is mainly due to the fact that the TCP implementation misinterprets packet loss due to mobility as an indication of congestion in one or more network resources. They also proposed a simple TCP protocol modification which uses the network layer information of any ongoing mobility to extend the slow-start phase in the recovery process. The authors showed that the modified TCP protocol significantly improves the throughput for host limited TCP connections in the case of a typical 0.15 seconds handoff interruption for every 120 second. There is a 20 percent increase in the case of large window sizes.

Another performance analysis of two TCP implementations in a wireless environment can be found in [8]. In this paper, the authors studied the protocol I, which is similar to TCP-Tahoe except it uses a fine-grained timer, and protocol II, which is a variant of TCP Vegas [26], in a simulation model with the similar characteristics similar to the measurement testbed in [7]. In the simulation, periodic and exponential intermobility times with a high bit error rate were considered. From the simulation results, the authors showed that both protocols performed, in general, badly when the wireless channel was not available 50% of the time, but protocol II was more robust than Protocol I when the wireless channel was available more than 50% of the time. The authors also concluded that the larger segment size gave slightly better throughput when there were no random errors but significant improvement in throughput at 5% random packet error rate.

The I-TCP approach described in [13] involves splitting the TCP connection into two

separate connections at the base station, one between the mobile host and the base station and the other between the base station and the mobile host. The advantage of this approach is to separate the flow and congestion control of the wireless links from that of the Internet; however, there are also many disadvantages of this approach which are the semantics problem, the special socket system and the software overhead. Another approach is to have its own retransmission protocol implemented at the data link level; however, studies have shown that this implementation leads to further degraded performance in the wireless environment [14].

1.4 Objective

Most of the earlier studies on the performance of TCP used simplified TCP versions over either wireless or wired links modeled by a tandem network of queues with a single traffic type. This thesis focuses on the performance of three different TCP implementations in wireless networking environments interworked with the Internet. In order to have a better TCP throughput performance, optimization of TCP parameters for the heterogeneous wireless and wired network under different conditions is considered. In this thesis, two cases are considered: two remote LAN's linked by a satellite network interworked with the Internet; and a mobile host interconnected with an Internet fixed host by mobile radio link in a Cellular Digital Packet Data (CDPD) environment [33]. It is important to abstract clearly the characteristics of the Internet and both wireless networking environments in order to carry out a detailed analysis of their impact on different TCP implementations. A lot of mutually dependent parameters have a great influence on the performance of TCP in the networking environments; therefore, the ultimate choice of these parameters and their value will seriously affect the TCP performance for bulk data transfer.

The objectives of the thesis are:

- to develop a queueing network model with the empirical characteristics of the Internet, the broadband satellite link, and mobile radio links;
- to analyze the effect of some key TCP parameters of the different TCP implementations in the heterogeneous networking environments with different conditions;
- to find the optimal values of the key parameters for the TCP implementations in the respective networking environments; and
- to provide suggestions in modifications of TCP for better throughput performance in both HWW networking environments.

This work differs from previous investigations in the following important ways:

- Three full versions of TCP (the standard version of TCP, TCP Reno, and TCP Vegas) are investigated and analyzed for networks with satellite links and mobile links interworking with the Internet.
- The Internet model is based on the empirical characteristics of wide area TCP/IP conversations, data packet traffic, and end-to-end behavior.
- A two-state Markov model is used for modeling a fading satellite channel.
- In the mobile computing environment, a three-state continuous time Markov model is used for modeling a fast fading mobile radio channel with frequent handoff interruptions.
- The jointly optimal values for the key TCP parameters are presented in both networks with satellite links and land mobile links.

1.5 Outline of the Thesis

In chapter 2, an overview of different TCP implementations is presented and discussed. Chapter 3 provides an overview of the wireless data communications systems, which are a broadband satellite communications network and a land mobile radio computing network, and the generic Internet environment with the relevant network models. Chapter 4 presents the simulation results and the performance analysis of different TCP implementations in a satellite network environment interworked with the Internet. The optimization of the chosen TCP parameters for the TCP implementations in this broadband satellite network is also given here. Simulation results and performance analysis for mobile computing networks with the various TCP implementations are given in Chapter 5. The optimization of the chosen TCP parameters in this environment is also presented here. Finally, Chapter 6 summarizes all the findings for both types of HWW networks, and provides suggestions for future work on improving TCP throughput performance in the networks with wireless links interconnected with the Internet.

Chapter 2 Three TCP Implementations

In this chapter, an overview of the generic TCP concept is presented. Three different TCP implementations including the standard version of TCP, TCP Reno, and TCP Vegas are studied and discussed in detail.

2.1 The TCP Concept

TCP/IP protocol suite is a networking protocol suite and the combination of different protocols at various layers. Figure 2.1 shows the 4-layer network system for TCP/IP. This protocol suite allows different kinds of computers, running on different operation systems, to communicate with each other over the worldwide Internet. Each layer of the TCP/IP protocol suite has different responsibilities. The link layer mainly handle all hardware details of physically interfacing with different types of media that is being used. The network layer handles packet routing over the worldwide Internet. The transport layer provides a flow of data between two hosts for the application layer above. The application layer handles the details of the particular application and user processes.

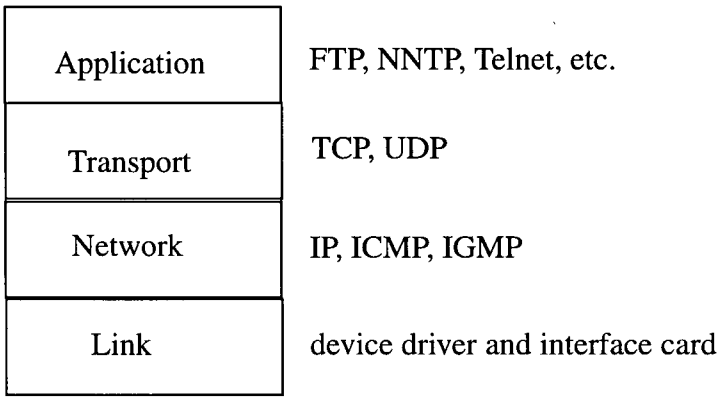

Figure 2.1 The four layer of the TCP/IP protocol suite.

Internet Protocol (IP) is a commonly use network protocol to route packets as a datagram by its routing mechanism. Since IP only provides a connectionless packet delivery service to route each packet separately, it has no guarantee of reliability or in-order delivery for each packet, which means that packet can be lost or destroyed by the media such as when network hardware fails, or packet can be substantially delayed by the dynamical network routing. Transmission Control Protocol (TCP), which is a commonly used transport protocol, has a responsibility for providing a reliable flow of data between two hosts. Therefore, the application layer can ignore all the details of the data reliability issue. Figure 2.2 shows the TCP data communications between two remote TCP hosts and all the protocols involved. TCP accepts files of any lengths from applications and breaks them up into smaller segments. Each TCP data segment is encapsulated in an IP datagram and sent from a source to a destination through the lower layers of the LAN's.

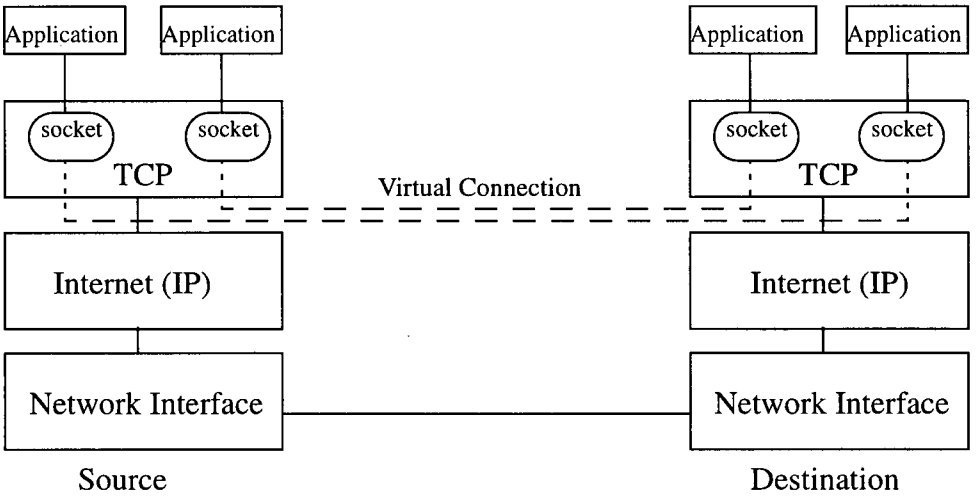

Figure 2.2 TCP Data communication between a TCP source and a TCP destination

TCP is a connection-oriented and end-to-end reliable transport protocol to support multi-network applications. In term of connection-orientation, two remote TCP applications must establish a TCP connection with each other before they can exchange data. Using a three-way

handshake mechanism with clock-based sequence numbers not only can avoid erroneous initialization of connections but also secure a unique end-to-end connection between two hosts. Since the specified sequence numbers are sent and acknowledged during the handshake, the handshake mechanism further guarantees that both sides are ready to transfer data and agree on the initial specified sequence numbers. When the connection is ready to close, TCP uses a modified three-way handshake to close the connection. TCP will close the connection in one direction when there is no more data to send, and then close the other half of the connection when it receives an acknowledgment that no more data is available.

In term of data reliability, TCP provides several main operations including basic data transfer, error recovery, and flow control. TCP, in general, decides when to block and send data at its own convenience to maintain a continuous stream of bytes in each direction between two end hosts. A stream of 8-bit bytes with no record markers inserted by TCP is exchanged across the TCP connection between two TCP applications. In this byte stream service, one end puts a stream of bytes into TCP, and the same and identical stream of bytes appears at the other end. Figure 2.3 and figure 2.4 show the encapsulation of TCP data in the IP datagram and the format of the TCP header, respectively.

TCP segments are used to establish connections as well as to carry data and acknowledgments. Each TCP segment is divided into two parts, the TCP header and the actual data. The TCP header only carries the expected identification and the control information. The "Source Port" and "Destination Port" fields identify TCP port numbers of the application programs at both ends of the connection. The "Sequence Number" field identifies the position of the segment in the sender's byte stream. The "Acknowledgment Number" field identifies the number of bytes that the source expects to receive next. The "HLEN" field contains an integer that specifies the length of the segment header measured in 32-bit multiples. The 6-bit "Reserve" filed is reserved for

future use. The 6-bit "Code Bits" is used by TCP software to determine the purpose and contents of the segment. The "Window Size" field is to specify the current receive buffer size on the other end. The "Checksum" field is used for error detection. The "Urgent Pointer" is used to specify the position in the window where urgent data ends. The "Option" field is used to negotiate with the TCP software on the other end of the connection.

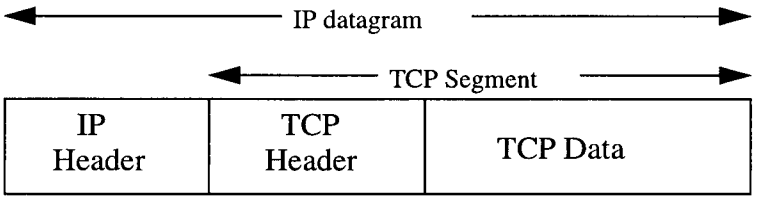

Figure 2.3 Encapsulation of TCP data in an IP datagram

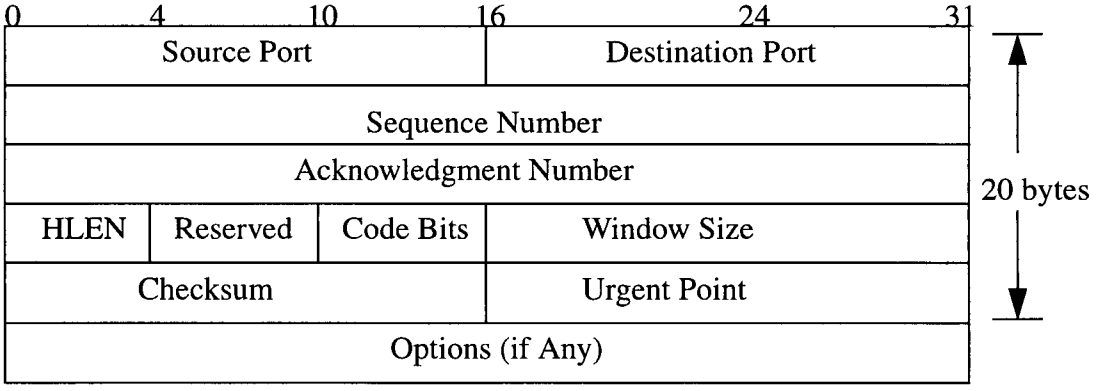

Figure 2.4 The format of a TCP header

In order to transport data reliably, a TCP sender uses a timer to maintain a time-out for a sending segment. A TCP receiver sends an ACK after it receives a correct data segment from the sender. If a corresponding acknowledgment from the receiver is not received in time, the segment is retransmitted. TCP also maintains a checksum on its header and data to check if there is any modification of the data in transit. It discards the damaged data segment and does not acknowledge receiving it. Moreover, it resequences the out of order data segments if necessary and passes

them in a correct order to the application. By using the positive acknowledgment, It can recover from data transmission errors such as damaged, lost, or duplicated packets over the unreliable Internet.

To prevent congesting a network and creating further losses, TCP provides efficient flow control. TCP senders uses a maximum allowable window size indicated in a received TCP header to control the amount of data that can be sent into the network. Congestion is a condition of severe delay caused by an overload of datagrams at one or more switching points. Therefore, a combined slow-start with congestion avoidance algorithm [17], referred as the slow-start mechanism, is implemented in TCP to further control the data traffic flow. The overview of the congestion avoidance and slow start algorithms are presented in section 2.2.1.

The operation of TCP can be explained with a finite state model. Figure 2.5 shows the finite state model of TCP, with circles representing states and arrows representing transitions between them. The label on each transition shows what TCP receives to cause the transition and what it sends in response. Applications must issue either a passive open command, waiting for a connection from another TCP host, or an active open command, initiating a connection to the destination. An active open forces a transition from the init state to the SYN_SENT state. After the three-hand shake mechanism (SYN_SENT and SYN_RCV), TCP moves to the ESTAB state and begins data transfer. All operations for data reliability such as the flow control and the error recovery perform in the ESTAB. To avoid having segments from a previous connection interfere with a current one, TCP moves to the TIME_WAIT state after closing a connection. It remains in these states (FINWAIT I and II) for twice the maximum segment lifetime before deleting its record of connection. To handle cases where the last acknowledgment was lost, it acknowledges valid segments and restarts the timer. Because the timer allows TCP to distinguish old connections from new ones, it prevents TCP from responding with a reset if the other end retransmits a

FIN request.

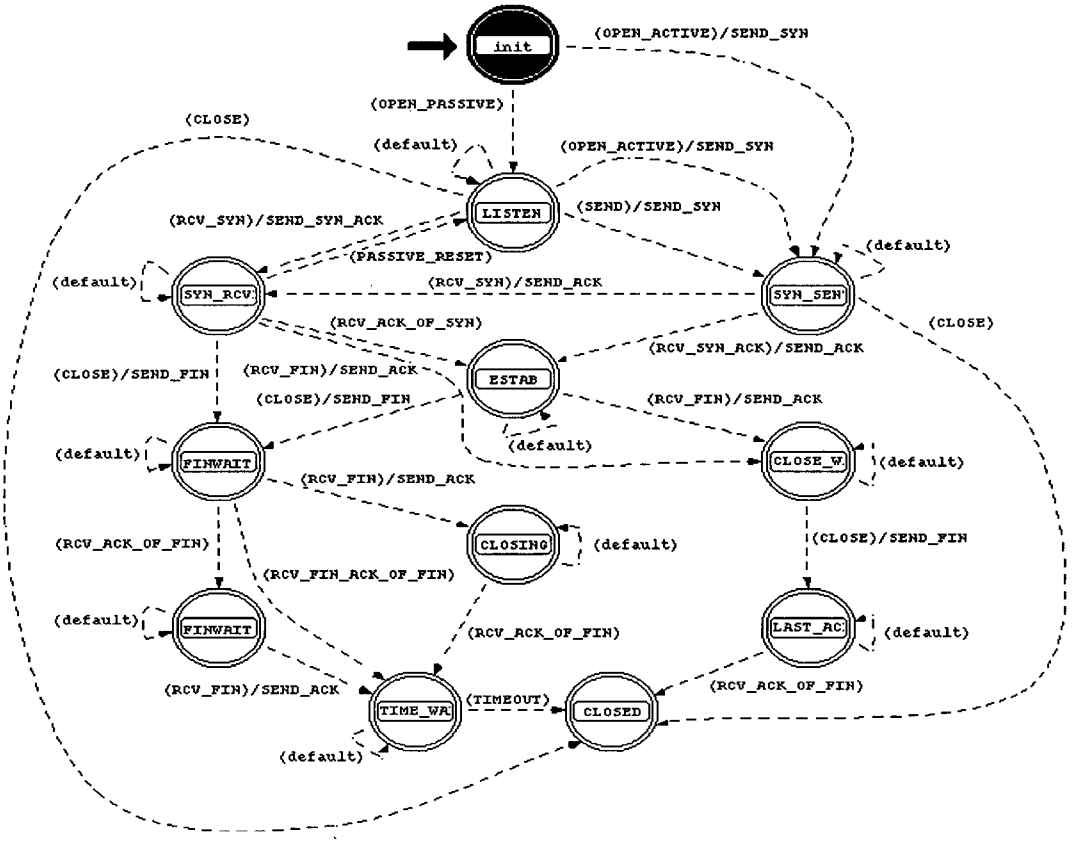

Figure 2.5 The TCP finite state simulation model

2.2 Three TCP Implementations

TCP described in [28] is referred as the standard version of TCP since it has been implemented in many commercial products over many years. Because the computer networking environments and the characteristics of the Internet traffic have been changing substantially, many options and modified TCP implementations have been proposed. However, the flow control and error recovery mechanisms are only slightly modified. Three versions of TCP studied here are the standard version of TCP (TCP RFC), TCP Reno, and TCP Vegas.

2.2.1 Standard TCP

TCP RFC was originally designed for a slow-speed, unreliable, datagram-based ARPANET where packets were often lost, duplicated or delivered out of sequence. In TCP RFC [28], the retransmission time out (RTO) algorithm is based on using the round trip time of each segment, a smoothing factor, and a delay variance factor to update a current round trip time (RTT) and to estimate a RTO for unacknowledged segments. RTT is measured by the elapsed time between sending a data byte with a particular sequence number and receiving an acknowledgment that covers that sequence number. A Smoothed Round Trip Time (SRTT) is computed as: SRTT = (α * SRTT) + ((1-α) * RTT) and used to update RTO calculated as: RTO = min [UBOUND, max [LBOUND, (β *SRTT)]], where UBOUND is an upper bound on the time-out and LBOUND is a lower bound on the time-out, α is a smoothing factor between 0.8 and 0.9, and β is a delay variance factor between 1.3 and 2.0.

Go-back-N ARQ scheme is used; therefore, if a segment is lost in transit, TCP RFC retransmits all the outstanding segments starting from the sequence number of the lost segment even through some of these packets may have been received correctly. In networks with high packet loss rates and long end-to-end delays, this mechanism significantly reduces TCP throughput because it may create more losses by further congesting the network.

Lost segments in the TCP congestion avoidance algorithm are used as a signal of network congestion. Congestion avoidance and slow start are independent algorithms with different objectives. They both require two variables, which are a congestion window, called *cwnd*, and a slow start threshold size, called *ssthresh*, to maintain the connectivity and the traffic flow between two end hosts. The congestion window is a second limit window employed by the slow start mechanism at the TCP sender to avoid network congestion. Whenever starting traffic on a new

connection, *cwnd* is initialized to one segment and *ssthresh* is typically initialized to 64 Kbytes. *Cwnd* is increased by one segment for every non-duplicated acknowledgment received. This provides an exponential increase in traffic flow. On a non-congested connection, the congestion window size in the sender is the same as the receiver window. The sender can transmit up to the minimum of the congestion window and the advertised window. Therefore, the congestion window is a type of flow control presumed by the sender while the advertised window is a type of flow control presumed by the receiver.

To estimate congestion window size when losses occur (indicated by a time-out or the reception of duplicated ACKs), TCP assumes most losses come from congestions and reduces its congestion window size. It saves one-half of its current window size in *sshtresh*, and *cwnd* is set to one segment only if congestion is indicated by a time-out, i.e., slow start. For those segments that remain in its allowed window, the RTO is backedoff exponentially, according to the exponential backoff algorithm. When new packet is acknowledged by the destination, increase in *cwnd* depends on whether TCP is performing slow start or congestion avoidances. If $cwnd \leq ssthresh$, TCP is performing slow start; otherwise, TCP is performing congestion avoidance. Slow start has *cwnd* beginning at one segment and exponentially increasing until it is halfway (i.e, equal to the new *ssthresh*) to where it was when congestion occurred, and then congestion avoidance takes over the flow control. Slow start exponentially increases *cwnd* by the number of ACKs received in a round-trip time after it is set to be one segment when time-out occurs. The increment of *cwnd* in congestion avoidance for every received ACK is estimated as equation (1), called linear increase, where $segsize$ is the maximum TCP segment size.

$$cwnd = cwnd + \frac{segsize^2}{cwnd} + \frac{segsize}{8} \qquad (1)$$

2.2.2 TCP Reno

TCP Reno is the most commonly used TCP implementation in today's systems. The RTO mechanism in TCP Reno [27] is refined by an RTT gain, an RTT coefficient deviation, and an estimator of an average of an RTT [17]. The measurement of an RTT in TCP Reno is the same as that in TCP RFC. A measured RTT for a particular sequence number is denoted by M. An updated RTO that is applied to M for the next transmitting segments is estimated as

$$Err = M - A$$

$$A = A + gErr$$

$$D = D + h(|Err| - D)$$

$$A = A + rttcoef \times D$$

where A, D, g, h, rttcoef, and Err represent, respectively, a smoothed RTT, a smoothed mean deviation, an average gain (set to 0.125), a deviation gain (set to 0.25), an RTT coefficient deviation gain (set to 4), and the difference between a measured value just obtained and a current RTT estimator. Therefore, a larger deviation gain for a deviation can make an RTO go up faster when an RTT changes. Karn's algorithm [11] is implemented to improve the TCP retransmission ambiguity problems. This algorithm specifies that when a time-out and a retransmission occur, the RTT estimators cannot be updated when the acknowledgment for the retransmitted data finally arrives.

The basic error detection mechanisms are based on a specified TCP segment timer maintained at a transmitter and the third duplicated ACK. A transmitter's system clock is recorded only when a selected segments are sent, and an RTO is calculated and updated only when an corresponding ACKs of these segments are received. The original retransmit time-out

algorithm is used, but checking for time-out occurs only when a coarse grain timer with a resolution of 500 ms has expired. A fast retransmit and fast recovery (FRFR) algorithm [17][18] is implemented to improve the original retransmission mechanism. This algorithm uses the third duplicated ACK as an indication of the need to retransmit a lost segment. When a transmitter receives three duplicated ACK in a row from a remote receiver, FRFR procedure immediately retransmits the earliest unacknowledged packet. This algorithm sets *ssthresh* to one-half of the current *cwnd*, retransmits the lost segment, and sets *cwnd* to the sum of the current *ssthresh* and three times of the maximum segment size. *Cwnd* is increased by one segment size for every duplicated ACK arrival; however, *cwnd* is only set to ssthresh when the next (non-duplicated) ACK arrives after an retransmission. TCP Reno still uses the combined slow start and congestion avoidance algorithm described in section 2.2.1 to control the data traffic flow into the network.

A selective repeat ARQ scheme is used; therefore, the request of retransmissions from the sender apply only for those packets that are not correctly received. A buffer is implemented in the TCP receiver to save and sort all the out-of-order packets. In the receiver, TCP saves the out-of-order received packets in the receiver buffer and responds with an ACK of the highest sequence number successfully received, plus one segment size. If the next received segments are also out of order, the data is saved and duplicated ACKs are generated. When the missing data segment arrives, the receiver now has in-sequence data segments starting from the missing data segment and passes these segments in order to the user process.

2.2.3 TCP Vegas

TCP Vegas described in [26] is a newly proposed TCP implementation which is based on TCP Reno. There are several modifications in retransmission, congestion avoidance, and slow-start mechanisms in TCP Vegas in order to improve throughput performance and decrease losses.

In the new retransmission mechanism, a finer-grained timer is used to calculate a more accurate RTT estimation. Its system clock is recorded each time a segment is sent, and the RTT is updated using the ACK arrival clock and its domestic clock recorded for the respective segment. Thus, TCP Vegas extends the FRFR algorithm implemented in TCP Reno.

By using this more accurate RTT estimation as a time-out value for a specified packet, TCP Vegas decides to retransmit segments only under two situations. First, when a duplicate ACK is received, it checks to see if the difference between the current time and the timestamp recorded for the relevant segment is greater than the time-out value for the relevant segment. If it is, then TCP Vegas retransmits the segment without having to wait for the arrival of the third duplicated ACK. Second, when a non-duplicated ACK is received after a retransmission, TCP Vegas again checks to see if the time interval since the segment was sent is larger than the time-out value. If it is, then TCP Vegas retransmits the segment. Using these two retransmission approaches, TCP Vegas can catch any other segments that may have been lost previous to the retransmission without having to wait for a duplicate ACK. It only decreases the congestion window if the retransmitted segment was previously sent after the last decrease. Coarse-grain time-out as in TCP Reno is still employed in case the above mechanisms fail to recognize a lost segment. The same selective repeat ARQ in Reno is used to retransmit the lost segments only.

In TCP Vegas, the congestion avoidance are based not only on dropped segments but also on changes in an estimated amount of extra data in the network. TCP Vegas defines the BaseRTT as the RTT of a data packet when the connection is not congested. It is the minimum of all measured round trip times. Under a good condition, the expected throughput should be equal to the size of the current congestion window (the number of bytes in transit) divided by the BaseRTT. The actual throughput is the number of bytes transmitted between the time that a specified segment is sent and its acknowledgment is received divided by the recorded sending

time for the distinguished segment.

TCP Vegas keeps the congestion window size unchanged when the difference between the expected throughput and the actual throughput is in the range of an upper and a lower threshold. Otherwise, it modifies the congestion window size by a linear increase or decrease as in TCP Reno. These thresholds are the multiple of a maximum segment size divided by the BaseRTT. The lower threshold allows the connection to utilize at least a certain number of buffers at a bottleneck router in the system, and the upper threshold ensures the connection uses no more than a certain number of buffers in the network. If the actual throughput is greater than the expected throughput, the BaseRTT is changed to the latest sampled RTT. When the difference between the expect throughput and the actual throughput, referred as Diff, is less than the lower threshold, TCP Vegas increases the congestion window linearly during the next RTT (see equation 1). When Diff is greater than the upper threshold, TCP Vegas decreases the congestion window linearly during the next RTT. In order to detect and avoid congestion during slow-start, TCP Vegas allows exponential growth in the congestion window only every other RTT. In between, the congestion window stays fixed; therefore, a valid comparison of the expected and actual throughput can be calculated.

2.2.4 Comparison and Summary

TCP RFC and TCP Reno have a similar congestion control mechanism except for the FRFR mechanisms implemented in TCP Reno. They have no mechanism to detect the onset of network congestion before packet losses occur, and thus they create their own losses due to buffer overflow by continually increasing window size. It is because the segment sending rate during slow-start is always much higher than the available bandwidth when using the exponential growth in the slow-start mechanism. Moreover, they only allow one segment to be sent when starting or

restarting after packet losses. Compared with both TCP Reno and TCP RFC, TCP Vegas has much better congestion control strategies to control the traffic flow. The congestion avoidance actions of TCP Vegas are based on changes in the estimated amount of extra data in the network to maintain a steady data flow into the network without creating its own losses.

Because of the fast retransmission and fast recovery mechanisms, TCP Reno has better retransmission strategies than TCP RFC. The time interval between sending a segment which is lost and retransmitting the lost segment in TCP Reno is much shorter than in TCP RFC. Moreover, the selective repeat ARQ in TCP Reno can significantly reduce the number of unnecessary retransmissions. Therefore, the performance of TCP Reno is much better than that of TCP RFC. TCP Vegas further modifies the retransmission mechanism in TCP Reno by using a more accurate segment retransmission time. The time to retransmit a lost packet in TCP Vegas is much faster than that in both TCP Reno and TCP RFC which is the slowest one. In the round trip time estimations, TCP Vegas is more accurate than TCP Reno and TCP RFC because Vegas estimates RTT and updates the current RTO for every segment; however, the detailed calculations may cause excessive delays in low packet loss rate situations.

Chapter 3 Heterogeneous Wireless and Wired Networking Environments

This chapter provides an overview of wireless data communications systems and the Internet environment and described the relevant network simulation models. Two common commercially used wireless data communications networks, i.e., a broadband satellite network and a mobile data network, are studied here. Broadband satellite communications can provide communications coverage over a very wide area and interconnections for users at remote areas. Mobile cellular communications systems can provide wireless data communications with non-stationary hosts.

3.1 Broadband Satellite Communications

Geostationary (GEO) satellites are currently used for satellite communications because of their fixed relay station. A satellite network can directly relay frames between two remote LANs. The inherent advantages of frame relaying for LAN interconnections in a satellite network are simple access interface, statistical multiplexing capability, and relatively high access speed. However, because of the long distance between the satellite and the earth, the propagation delay between a source and a destination is about 0.27 sec for GEO satellite networks. If using the congestion avoidance and the error control methods in TCP, a sender must wait at least 0.54 sec before an acknowledgment for a transmitted segment is returned to the sender.

Another factor that may degrade the performance of a satellite communication system is the possibility of a high bit error rate (BER) over a satellite link. In digital satellite systems [31] the nominal BER should be in the order of 10^{-8} or lower for clear sky operation. If the BER is worse than the BER thresholds, usually about 10^{-3} or higher, the digital satellite link may lose

modem synchronization and cease operations; therefore, the satellite system is unavailable for service at that point. At the higher frequency bands employed by satellite systems, one of the main causes of BER degradation is rain attenuation [38]. Attenuation caused by rain along a long transmission path is one of the dominant causes of signal fading in a space communication link. Rain drops absorb and scatter wave energy, resulting in a reduction in amplitude and randomness in the phase of the received signal, thus causing increased BER. Therefore, rain attenuation seriously degrades the reliability and the performance of the satellite links. The probability of packet losses is directly related to the BER and the packet length. If the current congestion control mechanism in TCP is used, the packet losses due to unreliable satellite links are also treated as indications of network congestion.

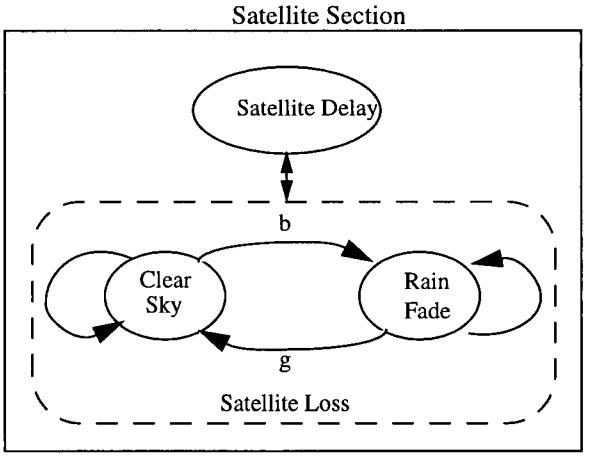

Figure 3.1 Broadband Satellite Model

3.1.1 Broadband Satellite Model

In order to take the effects of rain fades in satellite channels into consideration, packet losses over satellite links are modeled by a two-state Markov model with different BERs for the clear sky and rainfall conditions. Figure 3.1 shows the broadband satellite model which is composed of a satellite delay node and a satellite loss node. In the satellite loss node, the two-

state Markov model is composed of a clear sky state and a rain fade state. State transition is assumed to happen only after the transmission of each packet over the satellite. Errors in adjacent bits within a packet are assumed to be independent of each other for both forward and backward channels. The packet errors in this model are assumed to be randomly distributed.

The BER in the clear sky state and in the rainfall fade state are represented by "BER_C" and "BER_R", respectively. The loss probabilities of a data packet and an acknowledgment in the clear sky condition are P_{CD} and P_{CA}, respectively. The loss probabilities of a data packet and an acknowledgment in the rain fade condition are P_{RD} and P_{RA}, respectively. The transition probability from a clear sky state to a rain fade state is b, and the transition probability from a rain fade state to a clear sky state is g. The length of a data packet is L_D. The length of an acknowledgment is L_A. P_A is the average of the loss probability of an acknowledgment in the backward channel (see equation 2), and P_D is the loss probability of data packet in the forward channel (see equation 1). Since the P_A and P_D are independent, the average packet loss probability in the satellite channel, P in equation (3) presents the probability of a packet not being acknowledged because either the packet or its acknowledgment is lost.

$$P_D = \left(\frac{g}{b+g}\right)P_{CD} + \left(\frac{b}{b+g}\right)P_{RD} \qquad (1)$$

$$P_A = \left(\frac{g}{b+g}\right)P_{CA} + \left(\frac{b}{b+g}\right)P_{RA} \qquad (2)$$

Where

$$P_{CD} = 1 - (1 - BER_C)^{L_D} \quad , \quad P_{RD} = 1 - (1 - BER_R)^{L_D}$$

$$P_{CA} = 1 - (1 - BER_C)^{L_A} \quad , \quad P_{RA} = 1 - (1 - BER_R)^{L_A}$$

$$P = P_D + (1 - P_D)P_A \quad (3)$$

3.2 Mobile Cellular Communications

A basic cellular system architecture consists of three main parts which are mobile units, a cell site, and a mobile telephone switching centre. The mobile units have a transceiver and an antenna system. The cell consists of a control unit (base station), radio cabinets, antennas, and data terminals, providing interface between the mobile units and the swithching centre. The switching centre is the central coordinating element for all cell sites (radio base station sites), providing interface among different telephone company zone offices and controlling call processing. Cellular mobile systems are characterized by the enhanced capabilities of wireless terminals to exchange signaling with the remainder of the network through the switching centers or base stations.

Another main feature of the cellular mobile radio system is frequency reuse. In this frequency reuse system, users in different cells can simultaneously use the same frequency channel. The frequency reuse concept can be used in the time domain and the space domain. In the time domain, the frequency reuse occupies the same frequency in different time slots. In the space domain, the frequency reuse either can assign the same frequency in two different cells or repeatedly use the same frequency in the same general area in one system. The frequency reuse system can significantly improve the spectrum efficiency only if the system is properly designed and implemented.

Typical mobile wireless networks [6] [36] are composed of large numbers of base stations. Each base station has a coverage area and contains a channel which is used to establish,

maintain, update information, and terminate the connections of mobile hosts. The communication between the base station and the mobile hosts can be based on a single CDMA/TDMA channel. Cellular Digital Packet Data (CDPD) [33] is designed to operate as an extension of existing data communications networks as a peer multi-protocol, connectionless networks to existing data infrastructures. It provides a capability for 19.2 Kbps connectionless data packets layered on the top of the 30 KHz cellular phone channels and enables many data users to share one or more packetized data channels.

Because the antenna height of a mobile unit is much lower than its typical surroundings and its carrier wavelength is much less than the sizes of the surrounding objects, multipath waves cause severe fading at various rates of fluctuation. The signal fluctuates in general in a range of about 40 dB, thus causing a relatively high packet loss rate. Besides packet losses due to severe signal fading, packets may also be lost during handoffs. Handoff is a process of automatically changing frequencies as a mobile unit moves into different cells; therefore, the conversation can be continued in new cells without reconnecting/redialing. Because of handoff processes, communications may pause and all the packets may be totally lost during the handoffs between cells. Therefore, if the mobile unit moves fast, the rate of signal fluctuation is fast, and the probability of handoff is high, but the duration of fades is short.

3.2.1 Mobile Radio Model

In a real wireless environment, wireless channels may randomly become unavailable for a certain period of time due to movement across cell boundaries (depending on how fast the mobile unit moves and how large the handoff region is). The loss characteristics of wireless channels is mainly bursty due to various fading effects. Therefore, a error model with uniform error probability used for modeling the loss characteristics of the wireless channel seems likely to produce

inaccurate results. In order to capture the burstiness of wireless errors caused by the severe fading and the effects of mobility, packet losses over the wireless links are modeled by a three-state continuous-time Markov chain model based on the concept of the On-and-Off model [35]. The On-and-Off model assumes that the sources come on and go off for random intervals of time that are exponentially distributed with rate $1/T_{on}$ and $1/T_{off}$, respectively, where T_{on} is the average on time and T_{off} is the average off time for a source. Figure 3.2 shows the mobile radio simulation model.

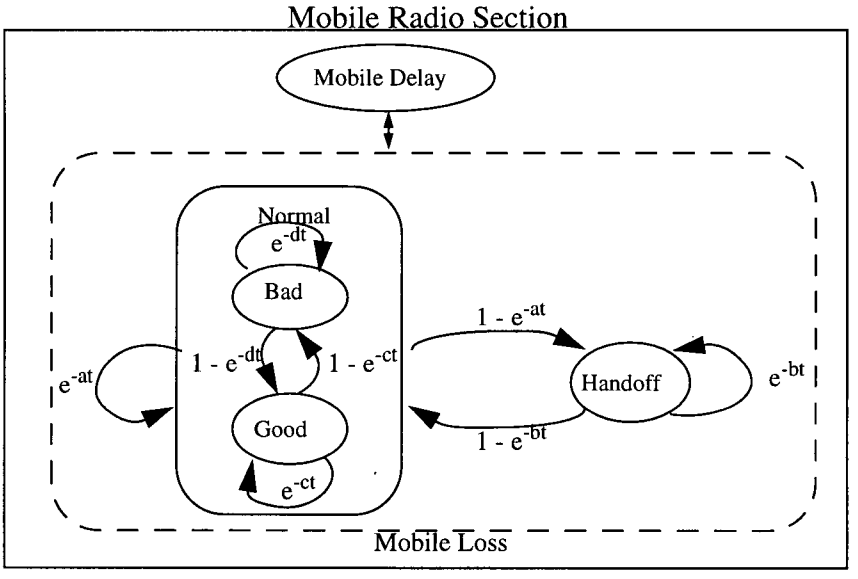

Figure 3.2 Mobile radio model

This mobile radio model is composed of a Mobile Delay node and a Mobile Loss node. The Mobile Delay node presents the average end-to-end delay for every packet transmitting over the mobile radio links, including the propagation delay and the transmission delay. In the Mobile Loss node, the three-state continuous-time Markov model is used to model the fast fading wireless channel which is composed of a handoff state and a normal state. In the handoff state, the end-to-end communications is totally stopped, which means that the connection is

momentarily lost due to handoffs between cells. All packets in this state are lost. Because of the bursty nature of wireless channel errors caused by severe fading, a two-state continuous-time Markov chain model (a good state and a bad state) is incorporated within the normal state. In the normal state, any sent packets would be corrupted with a relatively high BER when the channel stays in the bad state. When the channel stays in the good state within the normal state, any sent packets would be corrupted with a relatively low BER. The packet errors in this model are also assumed to be randomly distributed.

In this three-state model, each data communication alternates among states. The amount of time the channel spends in a state before making a transition into a different state is exponentially distributed with a defined mean. When the mobile unit moving within a cellular radio cell, the radio channel stays in the normal state. When the mobile unit moving within a handoff region, the radio channel stays in the handoff state. Figure 3.3 shows the occurrence of handoffs when a mobile unit moves into a different frequency radio cell.

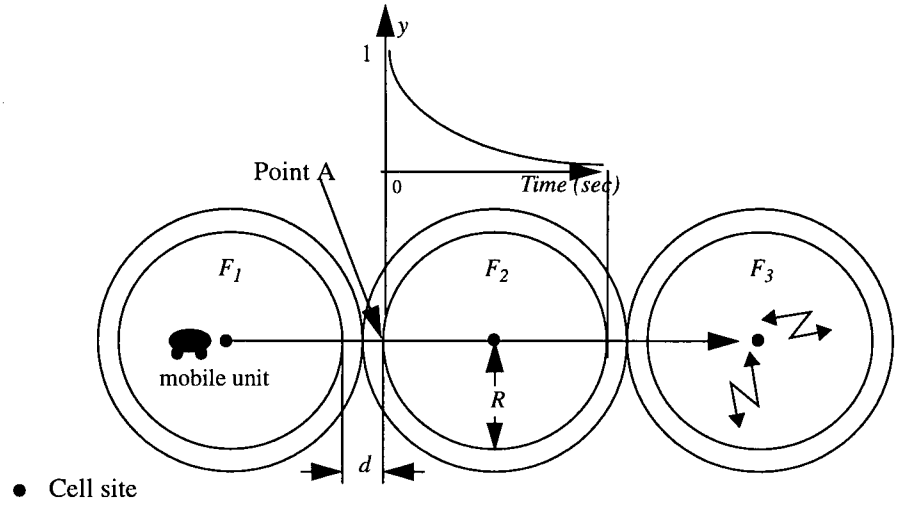

Figure 3.3 Occurrence of handoffs in a conventional cellular mobile system

Assume that a mobile unit with a constant speed randomly initiates a call in a radio cell

with a defined radius and unidirectionally moves from one radio cell to another radio cell. When the mobile unit reaches a cell boundary, it needs to pass a handoff region before it enters into a new radio cell. An average time of the mobile channel spending in a specified state (normal state or handoff state) is dependent on the size of the specified region and the mobile speed. In figure 3.3, 'R' represents the radius of a radio cell, 'd' represents the area of the handoff region, and 'F_z' represents the different frequency for a specified radio cell, where z = 1,2,3.

The normal and handoff periods are independent and exponentially distributed with mean durations of 1/a sec and 1/b sec, respectively, where 1/a is represented as the average time between handoffs and 1/b is represented as the average time performing handoffs. The estimation of the 1/a and the 1/b are given by equations (7) and (8), respectively. In equation (4), y represents the probability of the mobile unit staying in a specified state at time t, C is the initial value, and k is the constant of proportionality. At the initial point, t = 0, (for example, point A, right after the mobile unit entering into a new region), the probability of the mobile unit staying in the specified region (radio cells or handoff regions) is assumed to be 1.

Equation (5) shows the amount of time the mobile unit takes driving from a initial point (x = 0) to x in meters within the specified region, where v is the mobile speed. When the mobile unit reaches to half way of the size of a specified region (e.g. d/2 in a the handoff region), the probability of the mobile unit staying in the specified region is assumed to be 0.5. Therefore, the mean duration of the exponential distribution can be estimated as equation (6). From both equations (7) and (8), 1/a and 1/b are inversely proportional to the speed of the mobile unit and directly proportional to radius of the mobile unit and the handoff region, respectively.

$$y = Ce^{kt} \tag{4}$$

$$t = \frac{x}{v} \tag{5}$$

Therefore,

$$\frac{1}{k} \cong \frac{0.5x}{\ln(0.5) \times v} \tag{6}$$

When x = 2*R*,

$$\frac{1}{a} \cong \frac{R}{\ln(0.5) \times v} \tag{7}$$

When x = *d*,

$$\frac{1}{b} \cong \frac{0.5d}{\ln(0.5) \times v} \tag{8}$$

Within the normal state, the good and bad periods are also independent and exponentially distributed with means of 1/c sec and 1/d sec, respectively, where 1/d is the average duration of fades given by equation (9) and 1/c is the average duration between fades given by equation (11) in the steady state [36].

Average duration of fades:

$$\frac{1}{d} = \frac{\sqrt{2\pi}}{\beta v} \times \bar{t}_R \tag{9}$$

Level crossing rate:
(no.of fades per sec)

$$rate = \frac{\beta v}{\sqrt{2\pi}} \times \bar{n}_R \tag{10}$$

$$\frac{1}{c} = \frac{1}{rate} - \frac{1}{d} \tag{11}$$

In equation (9) and (10), β, ν, $\overline{t_R}$, and $\overline{n_R}$ are wave number ($2\pi/\lambda$), speed of the mobile unit, the normalized duration of fades with respect to *R* (distance measured from the transmitter to the receiver), and the normalized level crossing rate with respect to *R*, respectively.

3.3 The Internet Environment

The Internet has grown rapidly by several orders of magnitude in size in recent years. TCP is the most widely used Internet transport protocol, operating above the Internet protocol (IP) at the network layer, which can be characterized as being either host limited or network limited. The host limited network model is assumed to have infinite buffer size; there is no packet loss in the network, but the queueing delay, in general, is relatively large. The network limited network model is assumed to have finite bandwidth and limited buffer size; thus, packet losses are mainly due to congestion in one or more bottleneck network nodes. Therefore, the probability of network congestion and the queueing delay for every packet over the Internet are the dominant factors on TCP throughput performance. In order to have a more practical Internet model, the characteristics of the Internet traffic and the average delay over the Internet are considered. Figure 3.4 illustrates the simulation Internet model, composing of a Internet congestion node and a Internet delay node.

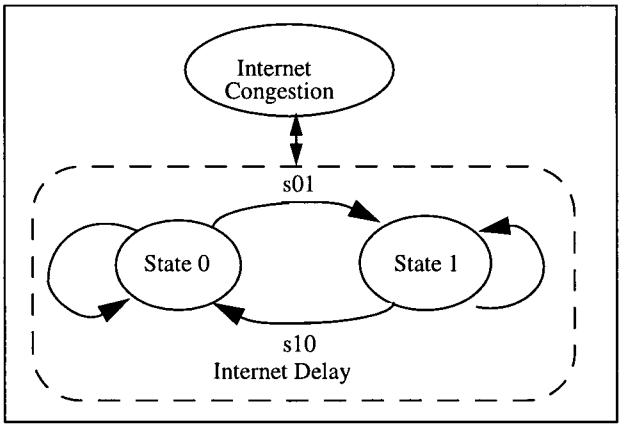

Figure 3.4 The simulation Internet model

3.3.1 Internet Congestion Model

Since TCP packets make up roughly 80% of all wide-area network traffic over the Internet, the characteristics of the TCP traffic can be represented as the Internet traffic characteris-

tics. In general, network traffic varies significantly not only in traffic mix but also in connection characteristics both over time and between different sites [23]. The wide-area network applications can be classified into two categories which are interactive traffic and bulk-transfer traffic. Interactive traffic is characterized as bidirectional small packet sizes ranging from 1 byte to 512 bytes. Over 90% of interactive conversations send fewer than 1000 packets and last less than a minute and a half. The ratio between the numbers of bytes sent by the originator and terminator of the interactive connection is about 1 to 20. Bulk-transfer applications are not strongly bidirectional and are request-response in nature.

In order to generate a random but realistic sequence of traditional internetwork conversations and data traffic for a set of sites, the preferred internetwork loading is bursty traffic which can be achieved by using different TCP traffic load models, as illustrated in figure 3.5. Without enforced synchronization, the simulation test scripts can generate a natural variation of traffic load during the long simulation duration. The author of [22] [24] found that most arrivals of TCP application conversations appear well-modeled as Poisson processes; however, it is not the case for all TCP data inter-packet arrivals. Most of the bytes transmitted by the TCP applications are sent by FTP, SMTP, NNTP, TELNET, and RLOGIN.

In the Internet traffic model used in the simulation (see figure 3.5), the only TCP applications included are FTP, SMTP, NNTP, TELNET, and RLOGIN. The arrivals of new TCP connection requests are modeled as time-varying Poisson processes with a specified site and time-of-day dependent rate. The TCP/IP wide-area characteristic library (tcplib) [23] is used to determine the characteristics of these TCP applications such as the duration of a specified TCP connection and the data size per connection. For interactive applications including TELNET and RLOGIN, the length of conversation in millisecond unit in the source is determined by the telnet_duration function in the tcplib. During the conversation, single byte packets are sent from the sender until

the time duration expires. The interarrivals between two source packets in the sender are obtained from the telnet_interarrival function in the tcplib. After receiving a single byte packet from the sender, the destination responds with a packet which size is determined by the telnet_pktsize function in the tcplib.

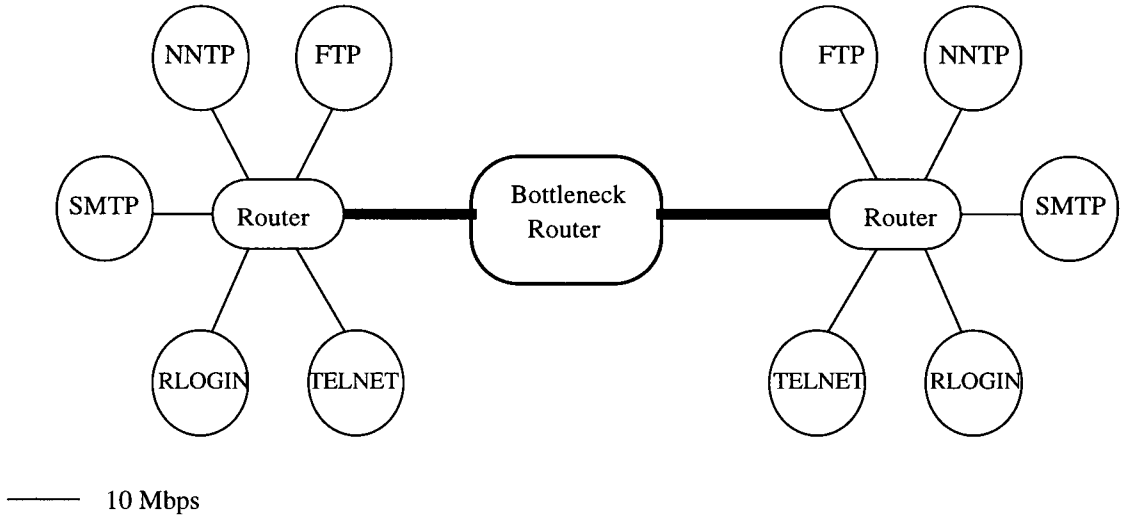

Figure 3.5 TCP application sources and Internet traffic model

For the bulk-transfer applications including FTP, NNTP and SMTP, the number of items sent per conversation in the sender and the number of bytes sent per item are determine by the specified functions in the tcplib. A source running FTP applications is simulated by using the ftp_nitems function in the tcplib to decide how many items to send. For each item, the ftp_itemsize function is used to determine the size of each item. Since the packet arrival process for a bulk-transfer data connection is largely determined by network factors such as the available bandwidth, congestion, and details of the transport-protocol congestion control algorithms, the distribution of the burst data sizes in bytes is quite heavy-tailed. Therefore, the packet interarrival times of these bulk-transfer are characterized by the Pareto Distribution. The handshaking between item transfers uses the ftp_ctlsize function in the tcplib to get the size of the handshake

packets. NNTP and SMTP applications are simulated in a similar manner as FTP applications; however, each SMTP and NNTP conversation transfers only one item.

Since the probability of the packet loss due to network congestion depends on buffer size, the bottleneck router in the Internet model is characterized as being network limited with a assumed buffer size of 30 Kbytes (a typical router buffer size). It is modeled as a store and forward queueing process model accepting packets from any number of sources and autonomously forwards them to a single destination module. The packet's service time may vary from packet to packet and is computed by dividing packet length, measured in bits, by the data rate of the T3 link. The conversation arrival rates of FTP, NNTP, SMTP, Telnet, and Rlogin are modeled as Poisson processes with rates of 42.3, 5.76, 3.27, 32.7, and 30.7 conservations per sec, respectively (measured arrival rate of UCB conversations at 10 am. [23]). Over a series of simulations, the probability of the network congestion in the bottleneck router in the Internet model is measured to be 0.05 ± 0.01 .

3.3.2 Internet delay model

In the Internet, packets are transferred as datagrams from their sources node to their destination node by the defined IP routing tables. Each datagram is handled independently from all other datagrams. This means that IP datagrams can be delivered out of order. If a source sends two consecutive datagrams to the same destination, for example packets A and B, each is routed independently and can take different routes, with packet B possibly arriving at the destination before packet A.

The observations of the end-to-end delay in the Internet [32] are that the round trip time varies substantially even over short periods of times and most of the losses occur one packet at a time. After a period of the dynamic changes of the round trip time, a step change behavior in

round trip time may remain effective for several seconds at a time, with only minor variations during this steady time. Therefore, a constant, exponential, or normal probability density function may either overestimate the end-to-end delay in the Internet or underestimate it to produce an inaccurate delay model. A better Internet delay model can be modeled by a sum of two functions [6], one of which is constant and the other is a random variable with an Erlang distribution. The constant term, which is the minimum network delay, can represent the average network delay in the case that no network congestion occurs and the queueing delay in each intermediate node is small. The variation of the network delay for a packet is dependent on the arguments of the Erlang distribution, which are a mean-outcome and an order of the distribution, as shown in equation (12). Equation (13) and (14) show the mean-outcome and the variance of the Erlang distribution.

$$f_x(x_0) = \begin{cases} (a^n x_0^{\,n-1} e^{-ax_0})/(n-1)! & x_0 > 0 \\ 0 & \end{cases} \tag{12}$$

$$E(x) = na^{-1} \tag{13}$$

$$\delta_x^{\,2} = na^{-2} \tag{14}$$

In equation (12), (13), and (14), n and a are the order and constant, respectively, in the condition of $n > 0$ when $a > 0$.

In order to capture most Internet end-to-end delay behaviors, a two-state Markvo chain model is used to model the Internet delay model (see figure 3.6). In state 1, a larger mean value of the Erlang distribution is used to generate a relatively long end-to-end delay. In state 0, a smaller mean value of the Erlang distribution is used to generate a relatively short end-to-end delay. The transition probability from state 0 to state 1 is $S01$, and the transition probability from state 1 to

state 0 is $S10$. Transition happens only after the transmission of each packet over the Internet. Based on the measurement results in [6] and [32], 96% of the packets stay in State 0. When a packet is in State 0, the packet has a shorter end-to-end Internet delay with minor variations. On the other hand, 0.04% of the packets stay in State 1. When a packet is in State 1, the packet has a longer end-to-end Internet delay, producing a few sharp rises within a few seconds.

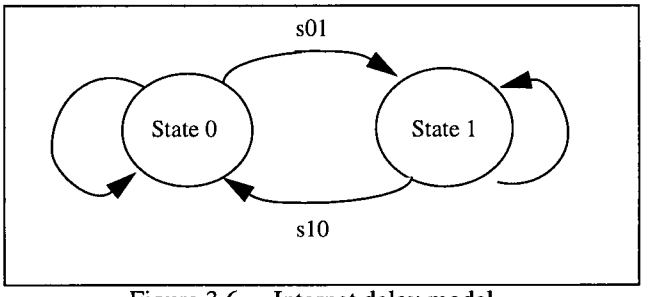

Figure 3.6 Internet delay model

By matching the means and the standard deviations of the first three sets of the experimental results in [32], Erlang distributions with an order of 2 is used for both states, the same order as suggested in [6]. A mean delay of 0.001 sec is used in State 0. 75% of the packets stays in State 1 with a mean delay of 0.01 sec and 25% of the packets stays in State 1 with a mean delay of 0.1 sec. The constant delay of 0.01 sec is used for the constant function in the delay model.

Chapter 4 Analysis of TCP in a Satellite Environment

In this chapter, the performance of three TCP implementations (TCP Vegas, TCP Reno, and TCP RFC) in an environment of a broadband satellite interworking with the Internet (BSII) is studied. Optimization of some key TCP parameters for the TCP implementations in the BSII environment is presented.

4.1 Satellite Network Interconnection

A schematic topology of a broadband satellite wide area network interworked with the Internet is given in Figure 4.1. A satellite channel is assumed to function as a transparent path between workstations on a remote LAN and its Internet gateway, communicating with other workstations on a different LAN over the Internet.

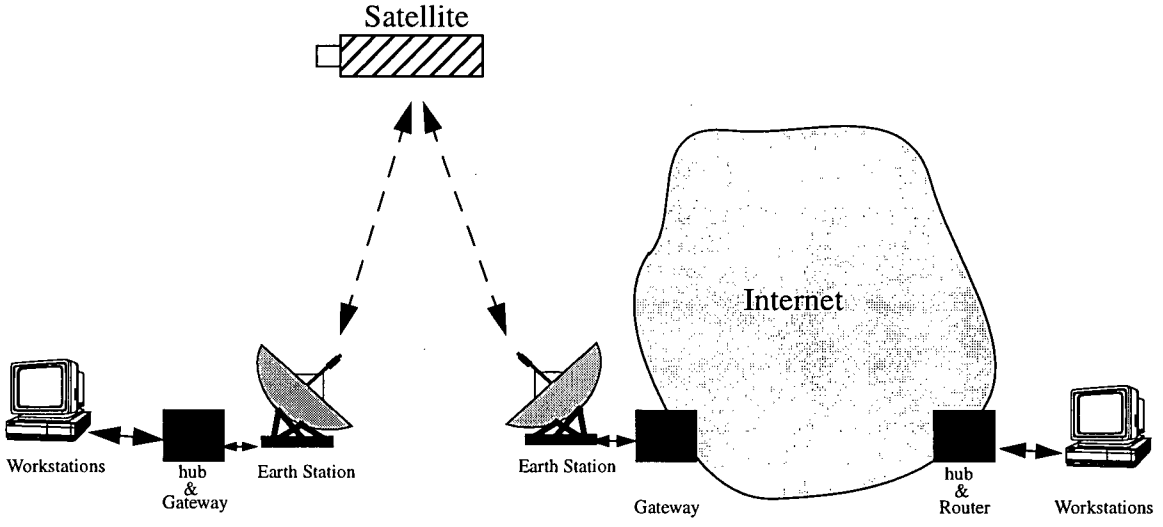

Figure 4.1 Satellite interconnected network topology

Propagation delay and transmission delay are two delay elements in the satellite channel. A gateway connects the satellite channel to the Internet. The TCP layer in one of the workstations

accepts bulk data files and breaks them up into smaller segments with a fixed maximum segment length for transmissions. The IP layer accepts the segments from TCP, appends an IP header to each segment, and sends each piece as a datagram. The datagrams are sent through the lower layer of the LANs and passed through the Internet. At the destination, TCP reassembles all the segments according to the sequence number assigned to each segment and recovers the original file. Aspect of communications related to the data link and the physical layer are not addressed in this thesis. In order to maximize TCP performance over the satellite network interconnected with the Internet, a number of TCP dependent parameters such as the maximum transmit window size (MTWS), the maximum ACK delay (MAD), and the Maximum segment size (MSS), etc., need to be jointly optimized under different conditions.

4.2 Simulation Model

To perform a detailed analysis of the behavior of different TCP implementations in the BSII environment, a sophisticated queueing network model for a typical Internet is developed. Packets are transferred from a source node to a destination node along paths and routers specified by IP routing tables. Figure 4.2 shows the simulation model for the BSII environment, including a number of nodes. The simulation model is divided into two sections which are the broadband satellite section and the Internet section. The gateway connects the broadband satellite section to the Internet section. The satellite section models the characteristics of the satellite channel described in section 3.1. The Internet section models the characteristics of the Internet described in section 3.3. In the simulation model, data packets are sent from the source node to the destination node, and the corresponding acknowledgment packets are returned immediately from the destination node to the source node after a fixed ACK delay. Passing through the Internet section and the broadband satellite section, the data packets and ACK are delayed and may be lost due to congestion or transmission errors in the media.

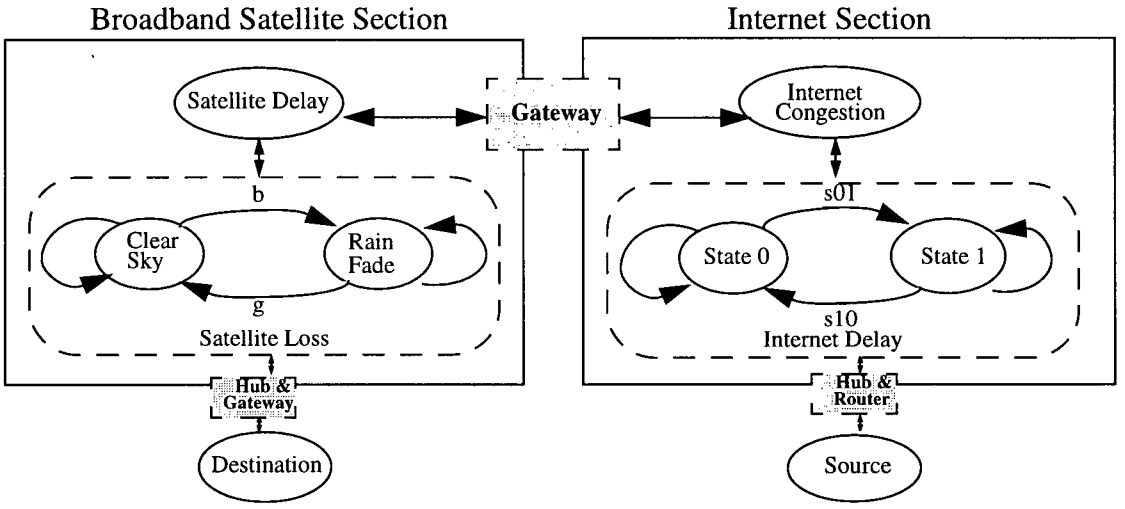

Figure 4.2 The simulation network model

4.3 Simulation Assumptions and Parameters

The different TCP implementations described in section 2.2, gateways, routers, the satellite section, the Internet section, the source and the destination are modeled by OPNET, a simulation tool. Since the big window option in [18] [21] can give a method of expanding the standard 64 Kbyte maximum window size to a 1 gigabyte allowable receive window size, the big window option is assumed to be used in all TCP end-to-end connections. File transfer is the only application considered for sending data from the source node through the Internet to the destination via a satellite channel. File transfer is performed over a logical connection that is opened between the transport layer at the two end nodes. Duplex transmission links are used in the simulation model. The satellite links are assumed to be lossy digital pipes subjected to transmission and propagation delays. The gateway and the routers function as a store-and-forward packet switching nodes with infinite buffer sizes in which packets are only subjected to queueing delay.

The only source of errors in the Internet section is the packet losses due to network congestion. Because of the serious effects of rain attenuation on a satellite link, the only packet

losses in the satellite channel are assumed due to rain fade conditions. These corrupted segments are detected by the checksum in the TCP layer and are recovered by the retransmissions. In the simulations several variable parameters including the packet loss rate (PLR) in the satellite channel, the average end-to-end (ETE) delay measured from the time the packet leaves the TCP layer in the source node to the time the packet is forwarded to the application in the destination, and the key TCP parameters including the MTWS, the MAD, and the MSS are considered. Table 4.1 summarizes the fixed simulation parameters for all the simulations in this chapter.

Table 4.1 Fixed simulation parameters for the simulation model

Simulation Parameters	
Header for IP datagram	20 bytes
Header for TCP segment with a big window option	22 bytes
Data rate of the LANs	10 Mbits/sec
Data rate of the Internet	45 Mbits/sec
Data transferred per run	30 Mbytes
*The average gain of the smooth RTT estimation	0.125
*The deviation gain of the smooth RTT estimation	0.25
The data rate of the satellite channel	1.544 Mbits/sec
The average one-way satellite delay	0.28 sec
The average congestion probability for the Internet model	0.04
The average end-to-end Internet delay	0.018 sec
The constant term of the Internet delay distribution	0.01 sec
The mean of the Erlang distribution for state 0 in the Internet	0.001 sec
The mean of the Erlang distribution for state 1 in the Internet	0.01 sec and 0.1 sec

* The average gain and the deviation gain of the smooth RTT estimation is used for estimating the retransmission time-out for the specified TCP segments.

4.4 Impact of Packet Loss Rate on TCP Throughput

This section presents the effects of the PLR due to the unreliable satellite links on the throughput of different TCP implementations. Satellite links are usually designed with low clear

sky BER; however, BER and hence PLR could increase substantially during rain fades. Frequency and duration of rain fades depend on the climatic region in which the earth station is located. Therefore, to determine the effects of the PLR in the satellite link on TCP performance under different conditions, the PLR in the rain fade state and the state transition probabilities (STPs)are varied in a series of simulations. All the key TCP parameters are fixed with values as suggested in [21][17]. The thresholds (α and β) in TCP Vegas are assumed to be 1 and 3. The RTT coefficient deviation is assumed to be 4. The average ETE delay is assumed to be 0.3 sec. Figure 4.3 compares the throughput performance of different TCP implementations as a function of the PLR in the satellite channe. In figure 4.3, state transition probability (STP) from clear sky state to rain fade state is assumed to be 0.01 (a upper bound percentage of the measured rain fade level [39]) and STP from clear sky state to rain fade state is assumed to be 0.05.

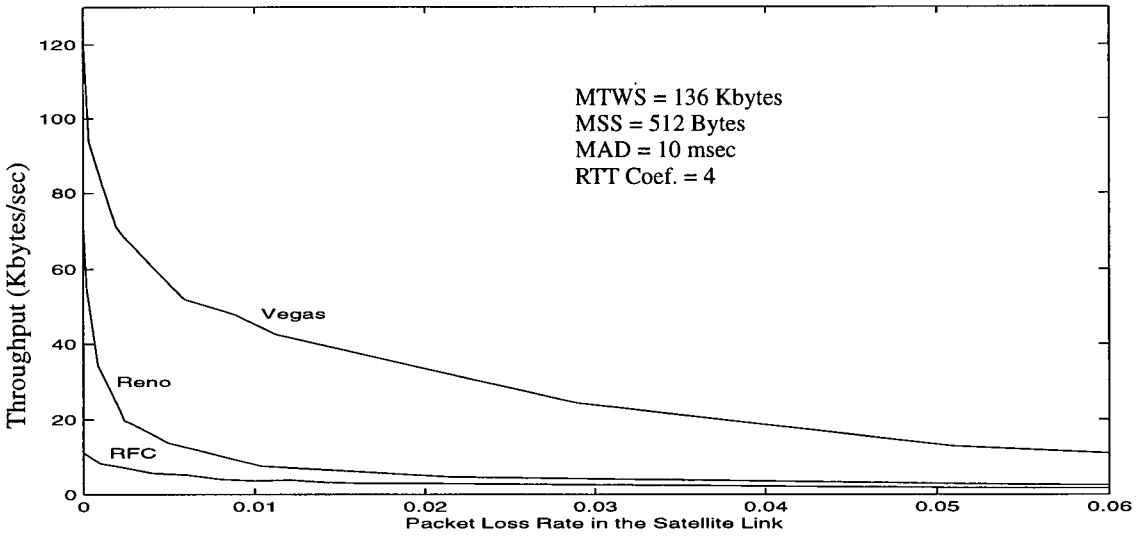

Figure 4.3 Throughput of different TCP versions vs. the PLR in the satellite link

From figure 4.3, because the slow-start mechanism treats packet losses over the satellite channel as indications of congestion, the throughput of the three different TCP implementations decreases significantly when the PLR in the satellite link increases. Because of the dynamic fluctuation of the congestion window, the long delay satellite channel cannot be fully utilized even

with the implementation of the big window option. In this satellite channel, TCP Vegas gives approximately four times better throughput than TCP Reno and six times better throughput than TCP RFC for Packet loss rates (PLRs) in the satellite links less than 0.03. Beyond the PLR of 0.03, the throughput of TCP Vegas comes closer to the throughput of both TCP Reno and TCP RFC. The throughput of both TCP Reno and TCP RFC do not change much when the PLR is increased beyond 0.02. Overall, the PLR has much greater effect on the throughput of TCP Vegas over the satellite channel than the other TCP implementations, and TCP Vegas has much better throughput than the others under the whole range of the PLRs.

Figure 4.4 plots the congestion window response for different TCP implementations versus time under a clear sky condition. It also shows a more detailed picture of the behavior of the congestion window for the TCP implementations, responding to the packet losses mainly due to the network congestion. Figure 4.5 and 4.6, respectively, plot the congestion window response for TCP Reno and TCP Vegas versus time under different STPs with a BER of $5x10^{-6}$ in the rain fade state. They show the congestion window's behavior for both TCP versions, responding to the packet losses not only due to the congestion but also due to unreliable satellite channel. For every packet loss, the congestion window size is reduced linearly by almost one MSS in TCP Vegas, by almost one half in TCP Reno, and to one MSS in TCP RFC.

In figure 4.4, because of its congestion avoidance approaches, TCP RFC has more sharp drops of the congestion window to one segment size than the other TCP implementations; whereas TCP Vegas has much less frequent drops of the congestion window than the others. However, because of its increasingly large congestion window and the exponential back-off algorithm, TCP Vegas seems to have longer communication pauses than the other two implementations if packet losses are detected by the time-out mechanism. This phenomenon appears as the flat and empty region of the curve indicated by "backoff period". Because the congestion

windows in both TCP RFC and TCP Reno are limited by Internet congestion, their throughputs are lower than the throughput of TCP Vegas even when there is no packet loss over the satellite channel.

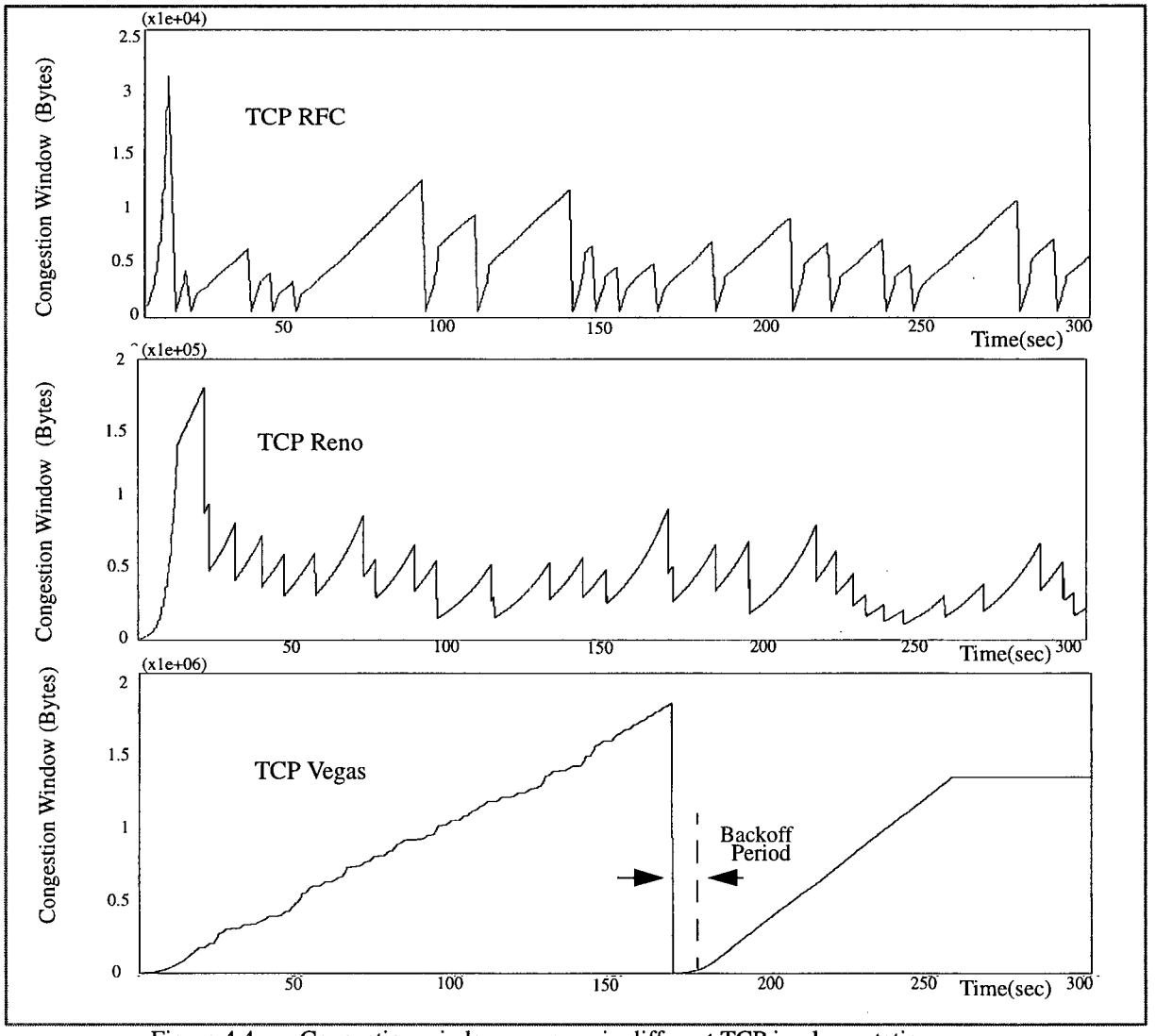

Figure 4.4 Congestion window response in different TCP implementations

In both figure 4.5 and 4.6, '*' represents the loss of a data packet and 'o' represents the loss of an acknowledgment frame. The STP from clear sky state to rain fade state is referred as 'b', and the STP from rain fade state to clear state is referred as 'g'. When b increases, the average duration of clear sky (between rain fades) is accordingly decreased, thus the satellite link

has a relatively high probability of staying in the rain fade state.

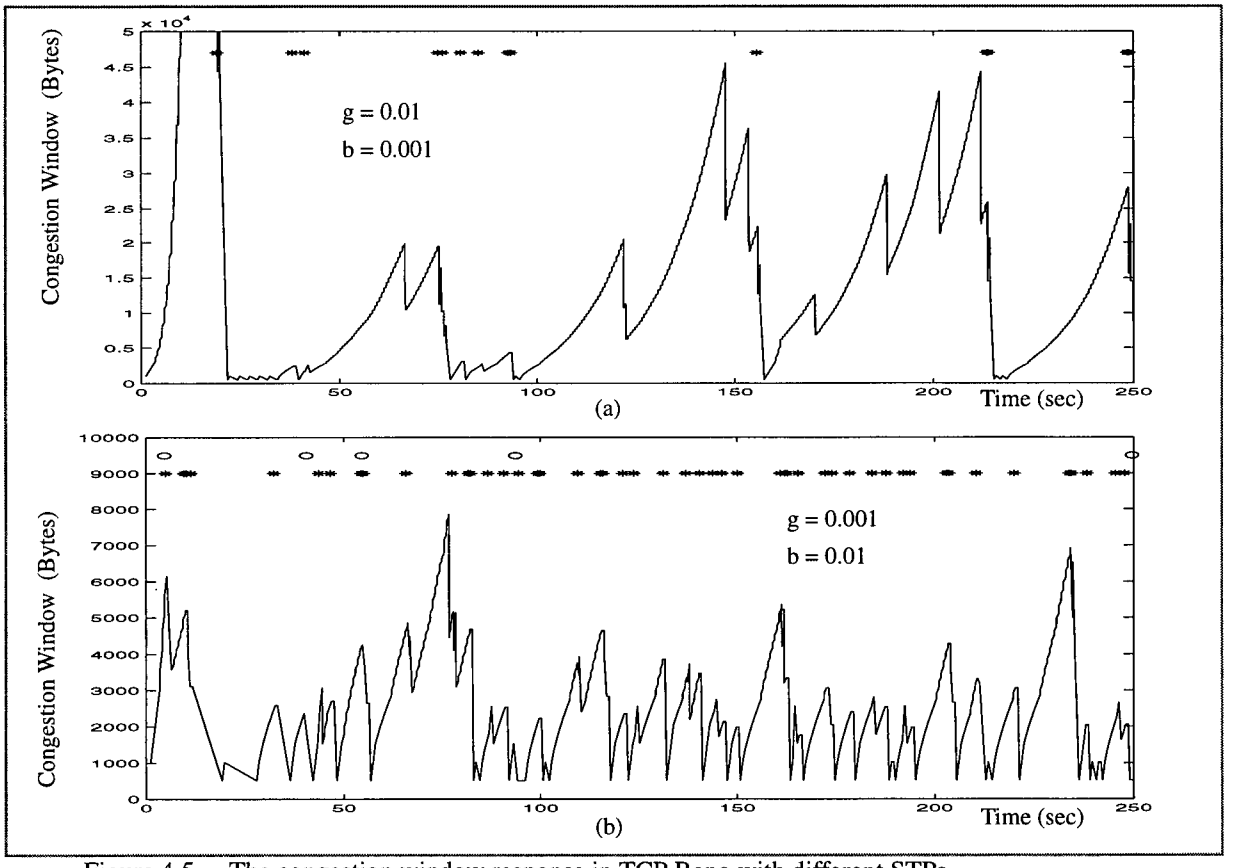

Figure 4.5 The congestion window response in TCP Reno with different STPs

Figure 4.5 shows how the congestion window responses in TCP Reno when data packets are lost within a full transmit window size under different STPs. In figure 4.5 (a), most of the retransmissions rely on the FRFR algorithm. At a higher b and a lower g (see figure 4.5 (b)), the average duration of each rain fade event (1/g) is relatively long; therefore, the probability of several corrupted data packets within a full transit window size is relatively high. The recovery of the connection in TCP Reno from a channel interruption is not fast enough to expand the current congestion window size before the next channel interruption arrival. Therefore, most of the retransmissions rely on the RTO mechanism and thus the congestion window size is always limited by this mechanism.

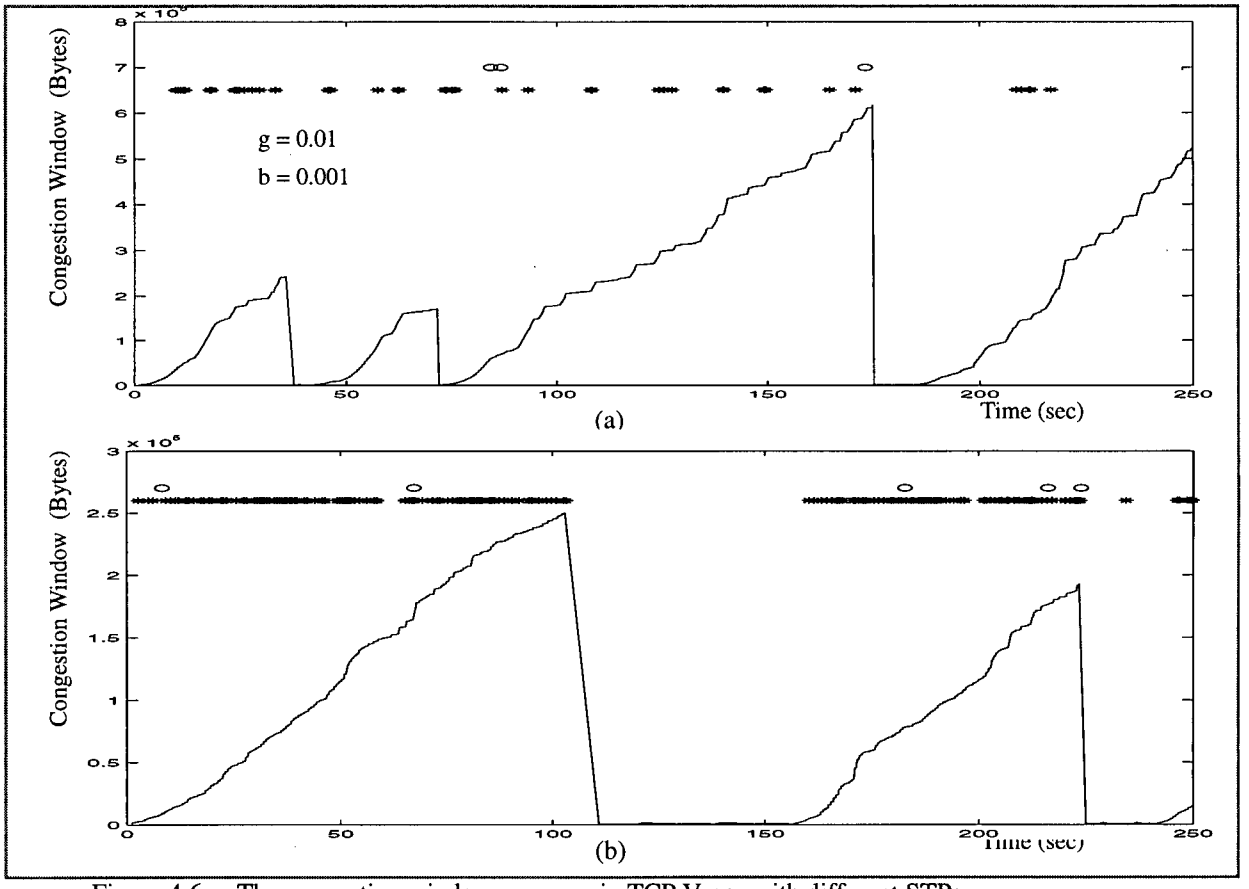

Figure 4.6 The congestion window response in TCP Vegas with different STPs

Figure 4.6 shows the congestion window response in TCP Vegas when there are several corrupted data packets within a full window size. Since TCP Vegas has a better error recovery mechanism in response to the numerous lost packets over the satellite channel, the congestion window size does not dynamically fluctuate even when b is increased and g is decreased. However, the current congestion window size is still limited by the increasing number of retransmissions. Because these channel interruptions due to the unreliable satellite link totally mislead the flow control mechanism in TCP Vegas, the congestion window is increased until when the fine-grain time-out mechanism in TCP Vegas fail to recognize a lost segment. Coarse-grain time-out as in Reno is employed.

In figure 4.6 (b), because of the larger congestion window and many lost packets within a

full transmit window size, the exponential back-off policy is extended and causes a long communication pause before the channel is recovered. Compared with TCP Reno and TCP RFC, TCP Vegas still has much better congestion avoidance to prevent losses due to congestion and has a much faster error recovery mechanism to quickly recover from any losses over the satellite channel. Therefore, the throughput performance of TCP Vegas, overall, is much better than the others for different PLRs over the satellite links.

4.5 Impact of End-to-End Delay on TCP Throughput

This section presents the effects of the ETE delay on TCP throughput. Because the satellite link does not have the high bandwidth of wired links and have higher latencies, there is substantially bandwidth changes between wireless and wired portions of the communications path. The queueing delay in the satellite system, which is the sum of the queueing delay in satellite modems and gateways between the wireless and wired portion of the communication path, is a critical element for this interconnection. An exponential distribution with a variable mean is used to represent the variations in queueing delay of the satellite system. In this section, all the network elements and TCP parameters are fixed and the same as those in section 4.6, but the average queueing delay in the satellite system is varied, and the average ETE delay is measured accordingly. Figure 4.7 compares the throughput performance of different TCP implementations as a function of the average ETE delay under a 0.01 PLR in the satellite channel and a clear sky condition. Figure 4.8 plots the sequence number versus time for the TCP data transfer over the satellite channel with a PLR of 0.001 versus time.

Figure 4.7 shows that the throughput efficiency of all TCP versions decreases linearly as the ETE delay increases under both clear sky and a PLR of 0.001 conditions. With increasing ETE delay, the sender must wait for a longer round trip delay for an ACK to be returned before it can send or retransmit packets into the network. As a result, the data sending rate in the sender is

significantly reduced and the error recovery period is lengthened, thus causing TCP throughput degradation. The ETE delay seems to have less effect on the throughput of TCP RFC than on that of both TCP Vegas and TCP Reno. Because of the faster error recovery and the better congestion avoidance mechanisms, TCP Vegas still has higher throughput than the other TCP versions over the whole range of the ETE delay even when the bandwidth delay product is large.

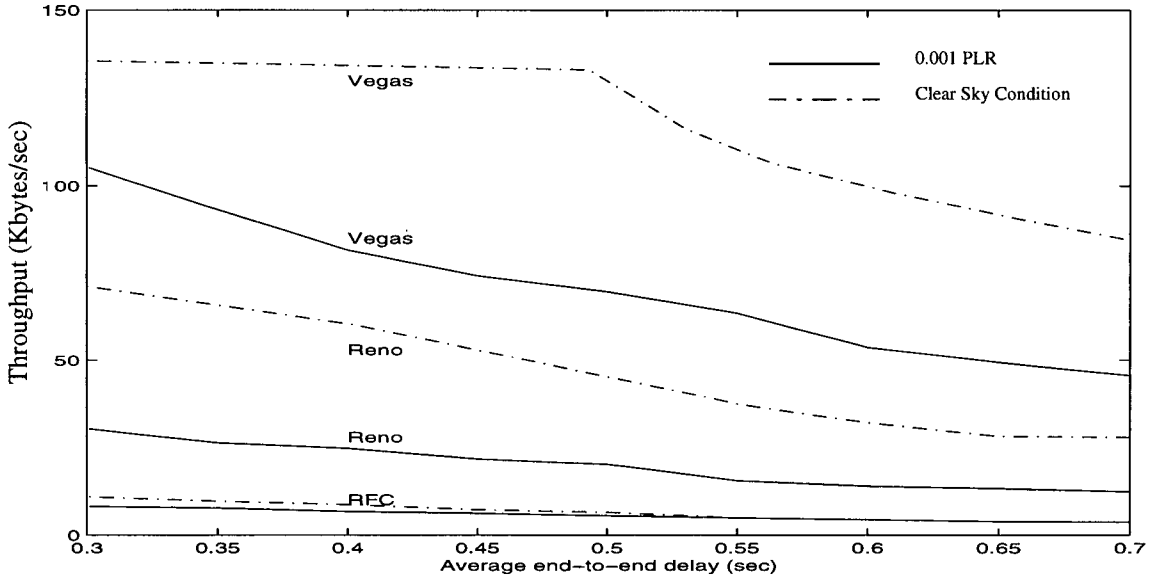

Figure 4.7 Throughput of different TCP versions vs. average ETE delay

Figure 4.8 shows a more detailed picture of the behavior of the ETE connection for different TCP implementations. The figure also shows the comparison of sequence number progression in a connection using TCP Vegas, TCP Reno, and TCP RFC implementations for different ETE delays. By increasing the ETE delay from 0.35 sec to 0.7 sec, there is approximately 40% reduction in the rate of the sequence number progression in all TCP implementations. As the ETE delay increases, the communication pauses become more often and each of the communication pause lasts much longer. This phenomenon appears as the empty regions of the curves. TCP Vegas maintains a relatively high and consistent throughput even at the ETE delay of 0.7 sec; however, both TCP Reno and TCP RFC unneccessarily invokes congestion control procedures

many times during the data connection.

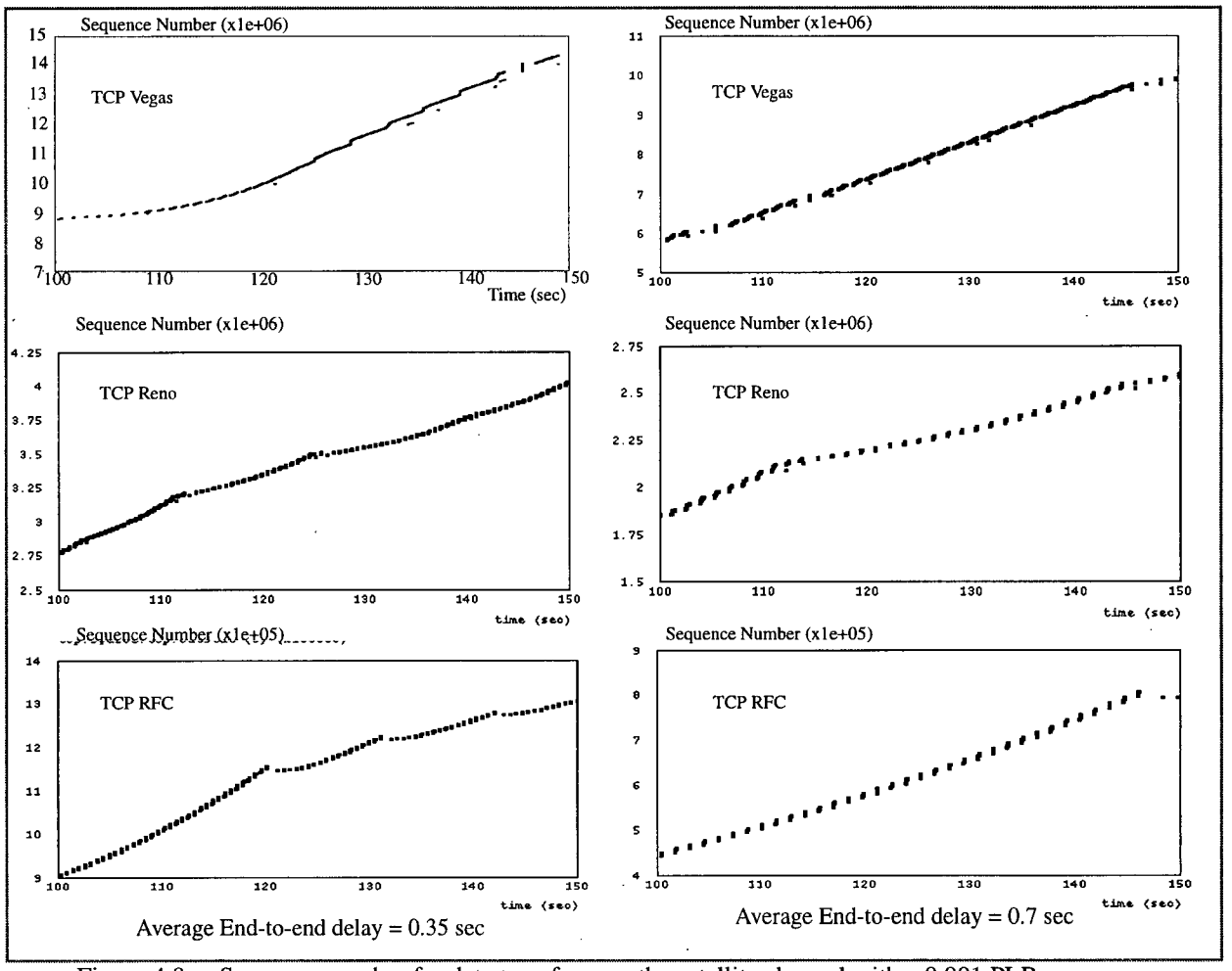

Figure 4.8 Sequence number for data transfer over the satellite channel with a 0.001 PLR

4.6 Impact of TCP Parameters on TCP Throughput

This section analyzes the effects of TCP parameters including the MTWS, the MAD, and the MSS on the throughput of the different TCP implementations over the satellite channel described in section 3.1. The MTWS and the MAD mainly control the rate of data transmission. The MSS not only controls the number of data bytes in transit per round trip time but also has a great effect on the PLR over the satellite links. These parameters strongly affect TCP throughput. If the values of these TCP parameters are too large or too small, the bandwidth efficiency of the

system may fluctuate over a wide range, thus causing TCP throughput inefficiency. Therefore, a detailed evaluation of these parameters is presented here. In this section, the thresholds (α and β) in TCP Vegas are assumed to be 1 and 3 [26], the RTT deviation coefficient is assumed to be 4 [17], b is assumed to be 0.01, and g is assumed to be 0.05.

4.6.1 Effects of Maximum Transmit Window Size

Many pervious studies in satellite interconnections [3][5][6] indicated that the MTWS strongly affects TCP throughput. In this subsection, the effects of the MTWS on the throughput of different TCP implementations are presented. All network components and TCP parameters are fixed for performance evaluation by only varying the MTWS. The MSS is assumed to be 512 bytes. The MAD is assumed to be 10 msec. The throughput of TCP RFC, TCP Reno, and TCP Vegas as a function of the MTWS for different PLRs under a 0.3 second average ETE delay condition are illustrated in figure 4.9, figure 4.10, and figure 4.11, respectively. Figure 4.12 and 4.13 show the throughput performance of TCP Reno and TCP Vegas as a function of the MTWS for different ETE delays under a clear sky condition.

By using the big window option [21], all TCP implementations can fully utilize the long satellite channel under no packet loss condition; however, the capability of this option is limited by the limited Internet resources. At a relatively low PLR, TCP throughput cannot be increased further and eventually drops when the MTWS increases beyond a certain value, indicating that the network resources are overloaded. Due to the strategies of the existing retransmission and congestion control mechanisms, all TCP implementations unneccessarily reduce the current congestion window size and over-activate the exponential back-off algorithm particularly in a relatively high PLR environment. As a result, their throughput performances are degraded and the benefits of the larger transmit window are nullified.

In figure 4.9, a MTWS greater than 16 Kbytes has nearly no influence on the throughput performance of TCP RFC for different PLRs because the Go-Back-N ARQ strategies is used for retransmissions. Beyond a certain MTWS for a specified PLR in the satellite link, TCP throughput seems to be constant over a wide range of the MTWSs. Therefore, there is no need for big window option in TCP RFC. Due to the activity of the slow-start mechanism at high PLRs, the congestion window size always be reset to one MSS, thus degrading extremely TCP throughput.

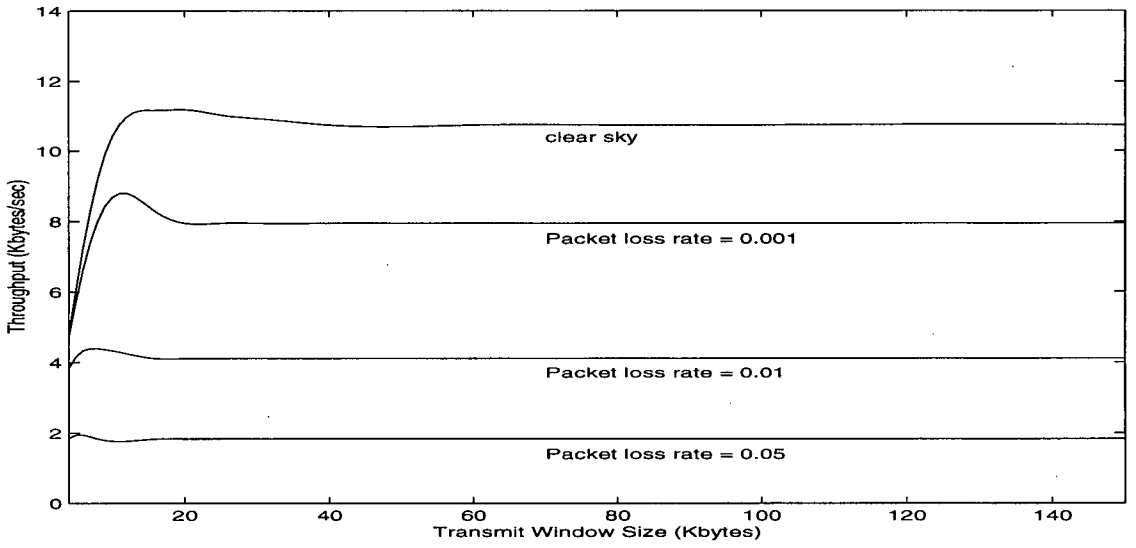

Figure 4.9 Throughput of TCP RFC vs. MWTS for different PLRs

Figure 4.10-4.13 show that the MTWS has greater influence on the throughput of TCP Vegas than on that of TCP Reno for different PLRs and different ETE delays. The big window option works well and greatly improves the throughput for both TCP implementations only at a relatively low PLRs and long ETE delays. Beyond an upper limit of the MTWS, the throughput of both TCP Reno and TCP Vegas decreases sharply for different PLRs and different ETE delays. It is because a bigger MTWS causes the sender to overflow the buffer in the Internet and creates more packet losses. At a higher PLR or a longer ETE delay under clear sky condition, increasing the MTWS beyond certain values seems to have no influence, particular in TCP Reno, on the throughput performance.

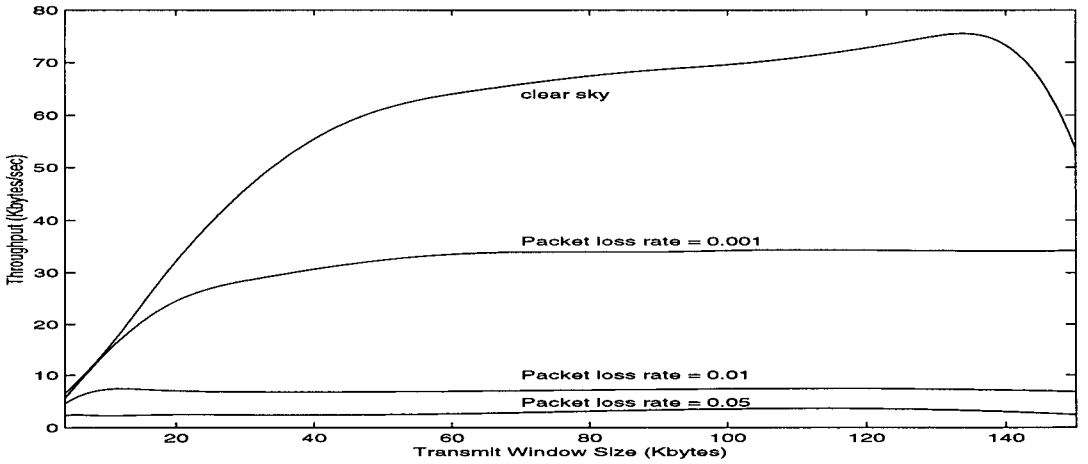

Figure 4.10 Throughput of TCP Reno vs. MWTS for different PLRs

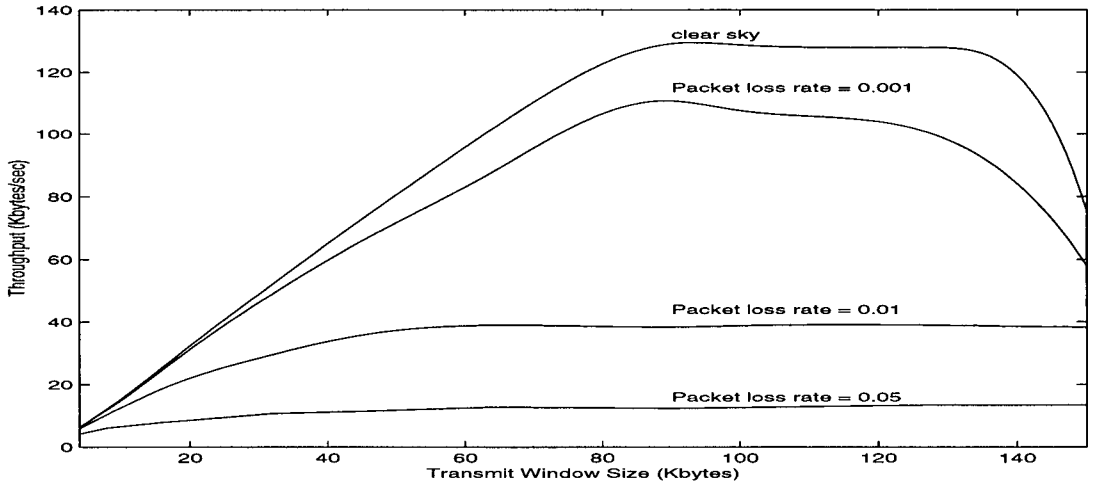

Figure 4.11 Throughput of TCP Vegas vs. MWTS for different PLRs

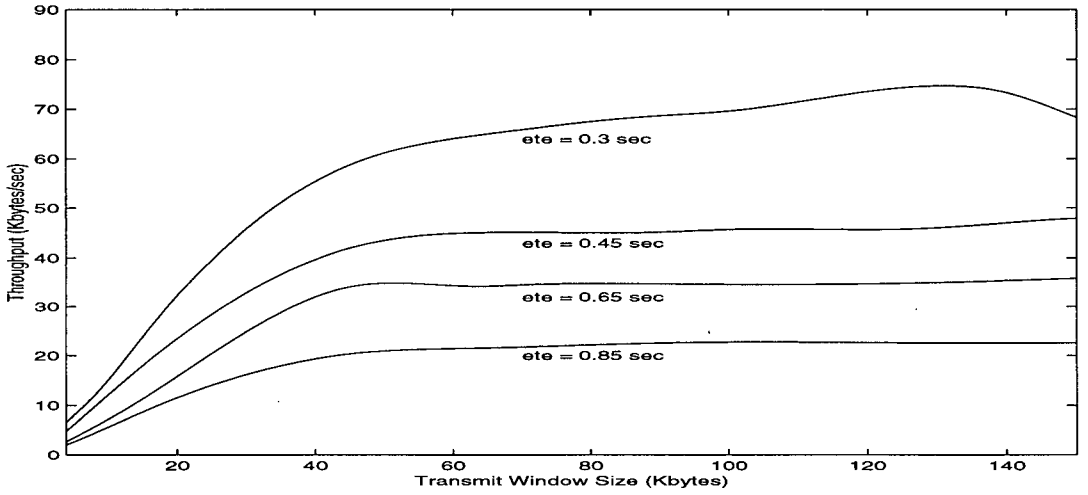

Figure 4.12 Throughput of TCP Reno vs. MTWS for different ETE delays

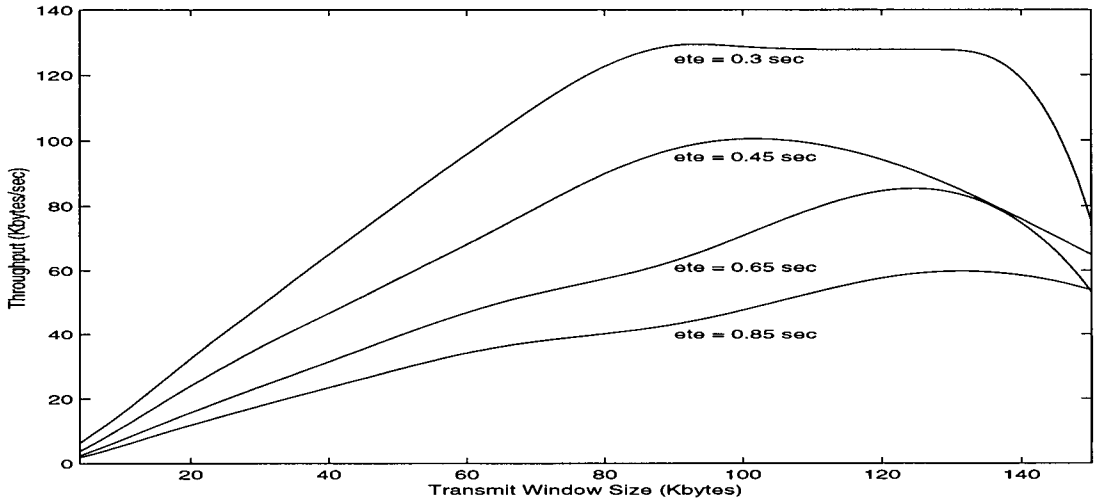

Figure 4.13 Throughput of TCP Vegas vs. MTWS for different ETE delays

The optimal maximum transmit window sizes (MTWSs) and their corresponding maximum throughput for different TCP implementations under different PLR conditions are summarized in Table 4.2. It seems that a bigger MTWS is suitable for TCP Reno and TCP RFC only at a relatively low PLR. At a high PLR, a much smaller MTWS seems to give a slightly better throughput for both TCP Reno and TCP RFC. TCP Vegas performs well not only at a relatively low PLR but also at a relatively high PLR. At a PLR of 0.001, TCP Vegas has approximately three times better throughput than TCP Reno and approximately thirteen times better throughput than TCP RFC. At the PLR of 0.01, TCP Vegas still has nearly six times better throughput than TCP Reno and nearly eight times better throughput than TCP RFC. Packet losses over the Internet cause the sender's congestion window size to dynamically fluctuate in both TCP Reno and TCP RFC, resulting in a small average transmit window size. Therefore, the throughput of both TCP implementation is limited by the Internet congestion. The use of a bigger MTWS in both TCP versions results in increasing packet losses in the Internet due to overload at the limited network buffers, and hence reduces their throughput performances.

With a better congestion avoidance mechanism, TCP Vegas can take an advantage of the big window option to send more data packets into the network without easily overflowing the

network buffer. With a faster error recovery mechanism, TCP Vegas can overcome the trade-off of the bigger MTWS to work reasonably well even at a high PLR. TCP Vegas with a bigger MTWS, overall, has much better throughput than the other TCP implementations in the HWW network; whereas, TCP RFC has worse throughput than the others under most conditions.

Table 4.2 The optimal MTWSs for different TCP versions under different conditions

	TCP Vegas	TCP Reno	TCP RFC
MTWS for the clear sky condition	128 KB	128 KB	24 KB
Throughput (Kbytes/sec)	78.52	130.15	11.89
MTWS at a PLR of 0.001	88 KB	64 KB	16 KB
Throughput (Kbytes/sec)	108.79	34.19	8.27
MTWS at a PLR of 0.01	64 KB	16 KB	8 KB
Throughput (Kbytes/sec)	42.76	7.26	4.39

4.6.2 Effects of Maximum ACK Delay

The sending rate and the error recovery period in the transmitter depend on the frequency of the ACK frames received. If the MAD is too small to allow out of sequence packet arrivals, unnecessary retransmissions will be increased; as a result, the data packet sending rate will be decreased, thus wasting a large amount of bandwidth. If the MAD is too large, the sender congestion window will grow too slowly, also causing low throughput performance. This subsection mainly studies the effects of the MAD on the throughput of different TCP implementations and presents the optimal maximum ACK delays (MADs) for different conditions. All the other parameters are fixed, but the MAD is varied for different PLRs and different average ETE delays. The MTWS is assumed to be 128 Kbytes. The segment size is assumed to be 512 bytes. The throughput of TCP RFC, TCP Reno, and TCP Vegas as a function of the MAD for different PLRs

under a 0.3 second average ETE delay condition are illustrated in figure 4.14, figure 4.15, and figure 4.16, respectively.

From figure 4.14, when the MAD is increased, the throughput of TCP RFC is gradually decreased and becomes constant. A smaller MAD gives higher throughput for TCP RFC at low PLRs. However, it seems that the MAD has a smaller effect on the throughput of TCP RFC at high PLRs. Figure 4.15 and 4.16 show that the throughput of TCP Reno and TCP Vegas increases as the MAD increases to a certain value for different PLRs, beyond which the throughput of both TCP versions is decreased. TCP Vegas with a bigger MAD seems to improve it throughput only for clear sky condition. Under high packet loss conditions, the MAD does not have great effects on the throughput of TCP Reno and TCP Vegas.

Figure 4.17 and 4.18 show the throughput of TCP Reno and TCP Vegas as a function of the MAD for different average ETE delays under a clear sky condition. At a smaller average ETE delay, the throughput of both TCP versions increases to a certain MAD and then gradually decreases. At a longer average ETE delay, the much smaller MAD seems to improve the throughput for both TCP Reno and TCP Vegas because a smaller MAD can reduce the error recovery period especially for the much longer bandwidth-delay products.

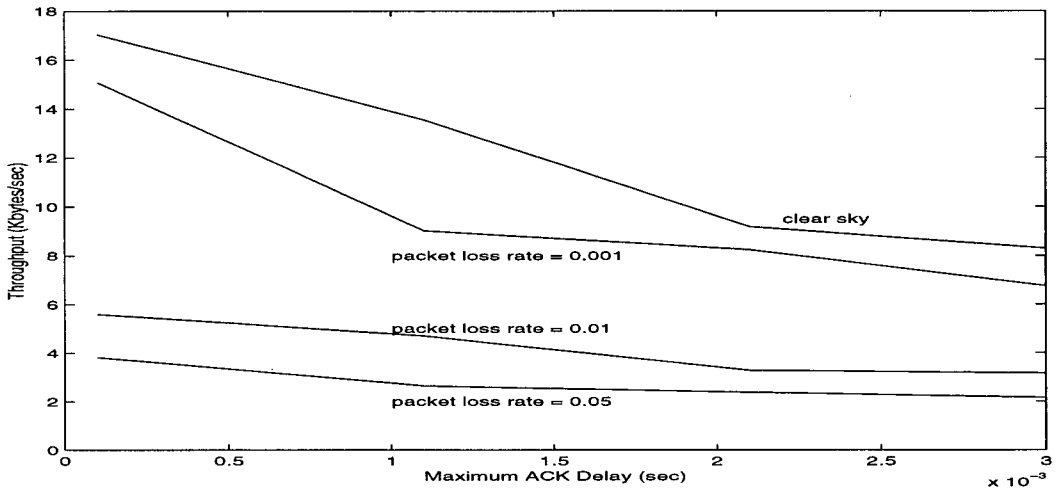

Figure 4.14 Throughput of TCP RFC vs. MAD for different PLRs

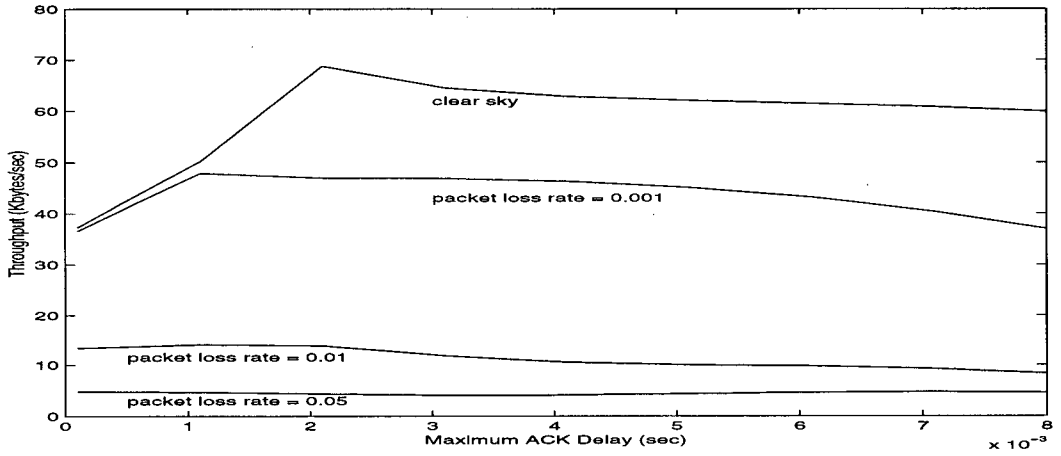

Figure 4.15 Throughput of TCP Reno vs. MAD for different PLRs

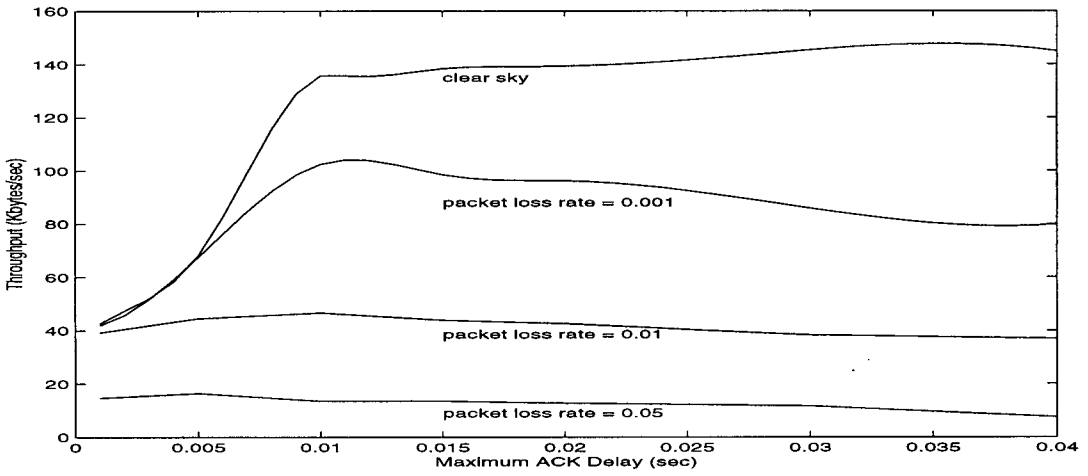

Figure 4.16 Throughput of TCP Vegas vs. MAD for different PLRs

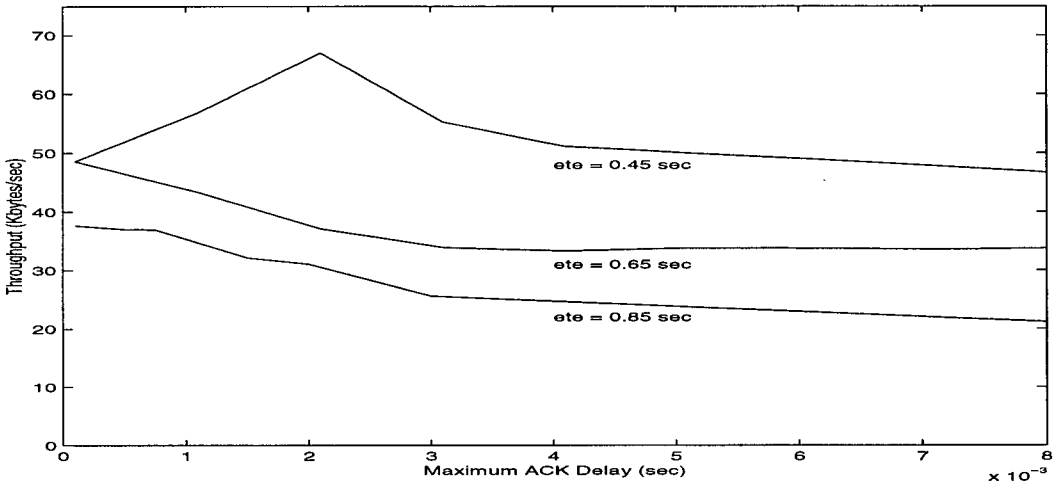

Figure 4.17 Throughput of TCP Reno vs. MAD for different ETE delays

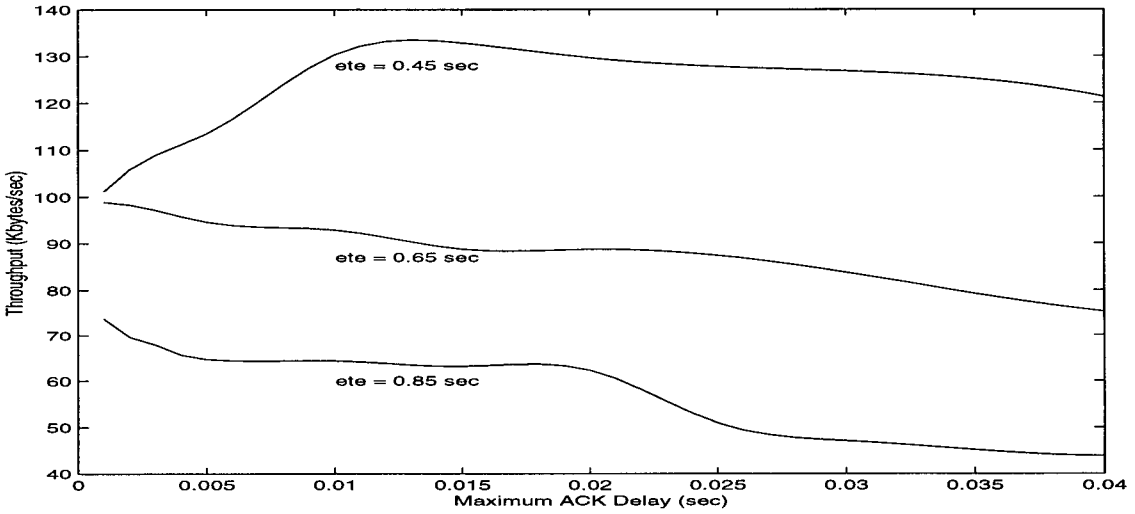

Figure 4.18 Throughput of TCP Vegas vs. MAD for different ETE delays

Table 4.3 The optimal maximum ACK delays for different TCP versions under different conditions

	TCP Vegas	TCP Reno	TCP RFC
MAD under a clear sky condition	20 msec	2 msec	0.5 msec
Throughput (Kbytes/sec)	145.47	66.76	17.67
MAD at a PLR of 0.001	10 msec	1 msec	0.1 msec
Throughput (Kbytes/sec)	102.55	53.48	15.12
MAD at a PLR of 0.01	10 msec	1 msec	0.1 msec
Throughput (Kbytes/sec)	46.54	15.57	5.82

The optimal MADs and the corresponding throughput for different TCP implementations under different conditions are summarized in Table 4.3. At different PLRs, a relatively long MAD seems to give higher throughput for TCP Vegas; however, a relatively short MAD seems to give higher throughput for both TCP Reno and TCP RFC. Because TCP Vegas uses its finer-grained timer to calculate a more accurate RTT estimation and update the current RTO for every segment, the accurate information in the returning ACK is very important. If the MAD is not large enough to wait for all out of sequence packet arrivals in the receiver, TCP Vegas in the

sender will receive the incorrect information from the returning ACK. As a result, it unneccessarily retransmits all the late arrival data packets in the receiver. Because of these misinterpretations, TCP Vegas may easily overflow buffers, thus causing more losses for the other connections which are sharing the same resources.

Because the retransmission of the lost packets in TCP Reno relies totally on the third duplicated ACK received and the retransmission of that in TCP RFC relies only on the RTO, the information in the returning ACK in both TCPs does not need to be as accurate as that in TCP Vegas. Moreover, the rate of the exponential growth of the congestion window size after a loss in both TCP implementations depends on the number of ACK frames received. Therefore, TCP Reno and TCP RFC prefer to use a relatively short MAD in order to quickly increase the data sending rate after a packet loss.

At high PLRs, all three TCP implementations prefer to have a shorter MAD in order to have much faster error recovery. However, at a low PLR, the TCP implementations prefer to have a relatively larger MAD in order to prevent the buffers from overflowing. At different PLRs in the satellite links and average ETE delays, the MAD has greater influence on the throughput of TCP Vegas than on that of TCP Reno and TCP RFC; whereas the MAD has the least influence on the throughput of TCP RFC. Overall, TCP Vegas has better throughput performance than the others over the whole range of the MAD under most conditions.

4.6.3 Effects of Maximum Segment Size

In order to obtain the maximum throughput performance for the TCP connection, the sender needs to send enough segments which should be as large as possible to fill up this long satellite pipe. Typically, the MSS has a slight effect on TCP throughput in the wireline networks because of the almost negligible packet lost rate due to the media. However, under a rain fade

condition, the average bit error rate is relatively high in a satellite channel, resulting in many corrupted packets. Since the PLR are directly depended on a specified BER and a specified segment size, the net effect of the segment size for the specific BER is to change the packet error rate, which effects on TCP have been considered in section 4.4. Equation (10) shows the relationship among the packet error rate, the segment size, and the BER. If the segment size increases at a specified BER, the PLR is almost linearly increased. Therefore, a smaller segment size is suggested for a relatively high BER environment in order to decrease the PLR [6] [15].

$$PacketErrorRate = 1-(1-BER)^{SegmentSize} \qquad (10)$$

However, it is not always true in the case of communications over a satellite channel with a relatively high STPs between clear sky and rain fade states. Typically, the satellite channel stays in the clear sky state for a while before the next channel interruption occurs. At a specified BER, the PLR in the satellite channel is not only dependent on the MSS but also on b and g (see equations 1 and 2 in section 3.1.1). At a higher b and a lower g, the probability of the satellite link staying in the rain fade state is relatively high. A larger segment size has a higher probability of error per packet than a smaller segment size particular in the rain fade state. Therefore, a larger segment size causes a relatively high PLR in the satellite link. However, at a reasonably low b and a reasonably high g, the probability of the satellite link staying in the rain fade state is relatively low. A larger segment size can carry more data information for every round trip time before the next channel interruption occurs. Therefore, a larger segment size may improve TCP throughput.

This subsection investigates the effects of the MSS on the throughput of different TCP implementations in the HWW environment over a range of the PLR in the forward satellite channel (P_D in equation 1). All simulation parameters are fixed, but the PLR in the satellite links and the MSS are varied. The MTWSs in TCP Vegas and TCP Reno are 128 Kbytes and the

MTWS in TCP RFC is 16 Kbytes. The MADs in TCP Vegas, TCP Reno, and TCP RFC are assumed to be 10 msec, 1 msec, and 0.1 msec, respectively. Figure 4.18, figure 4.19 and figure 4.20 show the throughput performance of TCP RFC, TCP Reno, and TCP Vegas, respectively, as a function of the PLR in the satellite channel for different maximum segment sizes (MSSs).

From those figures, the throughput of the different TCP implementations is increased when the MSS is increased from 512 bytes to 1458 bytes over a whole range of the PLR in the satellite links. It is because a larger segment size can carry more data information for every round trip time even through there is a trade-off of a relatively high probability of error per packet. However, the throughput of all TCP implementations is decreased when the MSS is increased from 1458 bytes to 1460 bytes because of the increase of the probability of IP fragmentation, as explained later, and a relatively high probability of error per packet. The MSS seems to have no effect on the throughput of TCP RFC beyond a PLR of 0.05. The effects of the MSS on the throughput of TCP Vegas is greater than that on the throughput of TCP Reno over the whole range of PLRs. At a relatively high PLR in the satellite channel, for example, PLR $\geq$ 0.03, the throughput of TCP Reno and TCP Vegas with a MSS of 1460 bytes is slightly better than that with a MSS of 512 bytes but much worse than that with a MSS of 1458 bytes .

It seems that a larger MSS can improve the throughput performance of all TCP implementations over a whole range of the PLR in the BSII environment. However, another limitation of TCP throughput performance is the maximum transfer unit (MTU) in byte of the interfaces. When the IP layer receives an IP datagram, It not only determines which local interface the datagram is being sent on but also queries that interface to obtain its MTU. IP compares the datagram size with the MTU and perform fragmentation if the datagram size is bigger than the MTU. When an IP datagram is fragmented, it is not reassembled until it reaches its final IP layer at the destination. In this case the MTU in a typical Ethernet LAN is 1516 bytes. Beyond this limitation of the MTU, the transit ethernet frame has a relatively high probability of

framentation. Subtracting the overheads of the ethernet frame and IP datagram, the MSS, in this case, is 1458 bytes.

A MSS of 1458 bytes seems to give higher throughput for all TCP implementations. With a MSS of 1458 bytes, TCP Vegas has 1.3 times better throughput than TCP Reno and nearly six times better throughput than TCP RFC at a relativlely low PLR. At a relatively high PLR, TCP Vegas has approximately two times better throughput than TCP Reno and five times better throughput than TCP RFC. Overall, TCP Vegas has much better throughput performance than the other TCP implantations for different MSSs over the whole range of PLRs in the satellite channel.

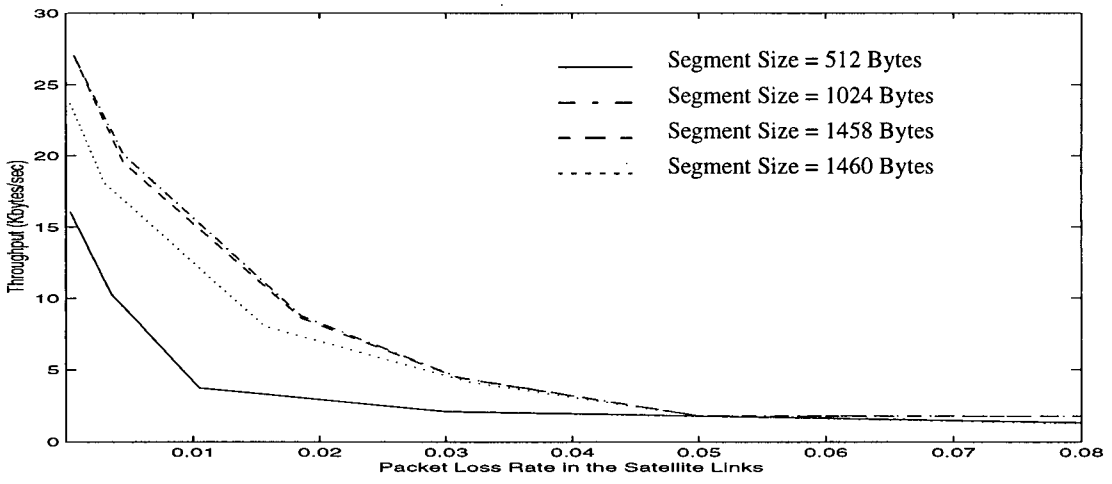

Figure 4.19 Throughput of TCP RFC vs. the PLR in the satellite links for different MSSs

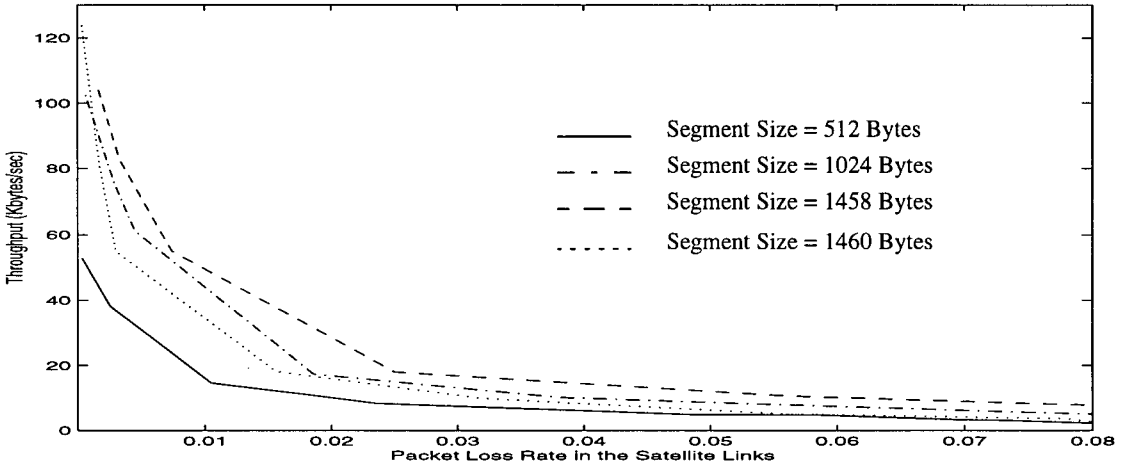

Figure 4.20 Throughput of TCP Reno vs. the PLR in the satellite links for different MSSs

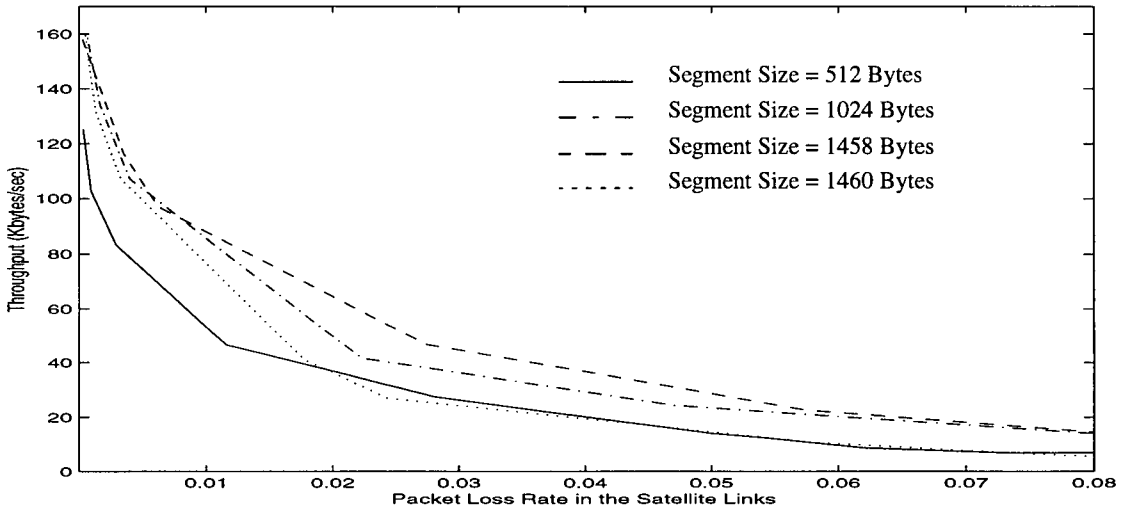

Figure 4.21 Throughput of TCP Vegas vs. the PLR in the satellite links for different MSSs

4.7 Optiminzation and Discussion

In order to improve throughput performance of different TCP implementations for a range of the PLRs in the satellite links, this section provides a detailed evaluation for jointly optimizing the TCP parameters including the MTWS, the MAD, and the MSS. From the simulation results in section 4.6, an upper bound and a lower bound of the TCP parameters are obtained and suggested for TCP RFC, TCP Reno and TCP Vegas under the assumptions in section 4.3 for different PLRs. In this section, all other simulation parameters are fixed, but the above TCP parameters are varied under different conditions. The RTT coefficient deviation is assumed to be 4 and the thresholds (α and β) in TCP Vegas are assumed to be 1 and 3. g is assumed to be 0.05. b is assumed to be 0.01. An average ETE delay of 0.3 sec is assumed in the first scenario and a PLR of 0.001 is assumed in the second scenario.

Since these TCP parameters do not have great effects on the throughput of TCP RFC, the throughput values of TCP RFC for different TCP parameter settings are too close to be statistically different. A bigger MTWS results in increasing packet losses in the Internet due to network

congestion. A smaller MAD and a larger segment size seem to improve the throughput performance for TCP RFC. Therefore, over the whole range of the PLRs, the optimal values of the MTWS, the MAD, and the MSS are suggested to be 16 Kbytes, 0.1 msec, and 1458 bytes, respectively in this HWW network.

Figure 4.22-4.24 show the throughput of TCP Reno as a function of the PLR in the satellite channel for different MSSs with different TCP parameter settings. From those figures, for different MTWSs and different MADs, TCP Reno with a MSS of 1458 bytes seems to have much better throughput than that with the other MSSs over the whole range of PLRs in the satellite channel. TCP Reno with a MSS of 512 bytes seems to have much lower throughput than that with the other MSSs. For different MTWSs and different MADs, TCP Reno with a MSS of 1458 has 2.5 times better throughput than that with a MSS of 512 bytes and has approximately 1.5 times better throughput than that with both MSSs of 896 bytes and 1024 bytes. However, TCP Reno with a MSS of 1024 bytes is just slightly better throughput than that with a MSS of 896 bytes.

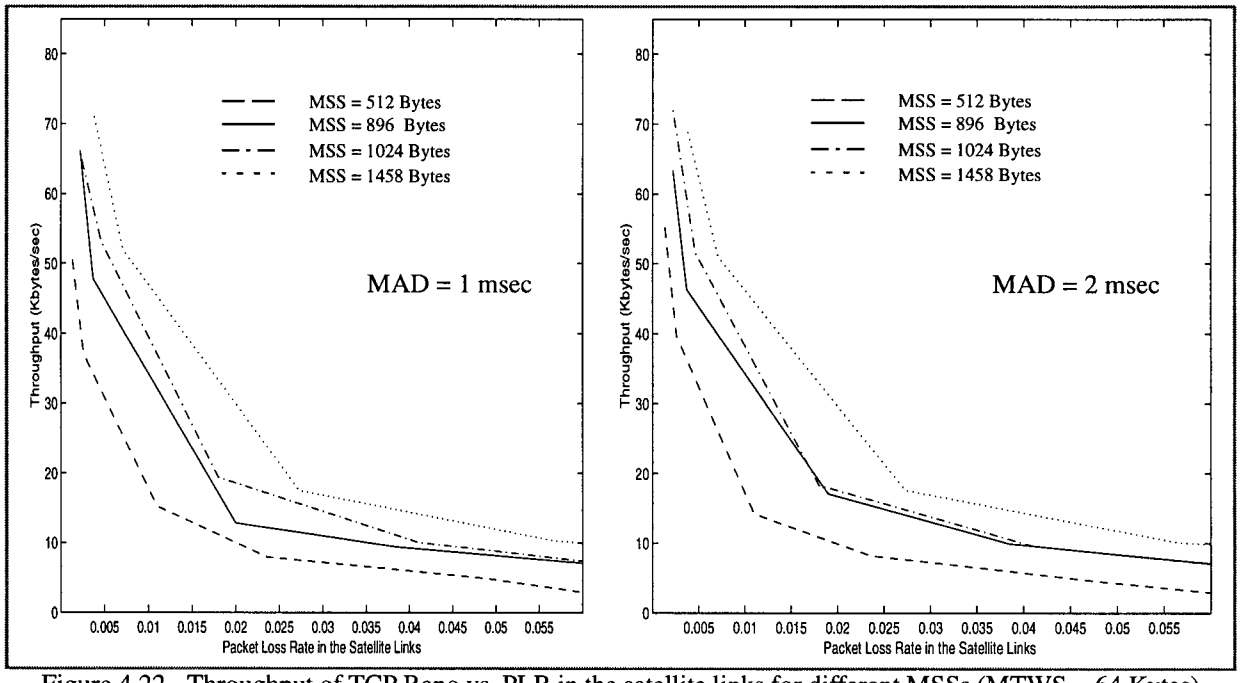

Figure 4.22 Throughput of TCP Reno vs. PLR in the satellite links for different MSSs (MTWS = 64 Kytes)

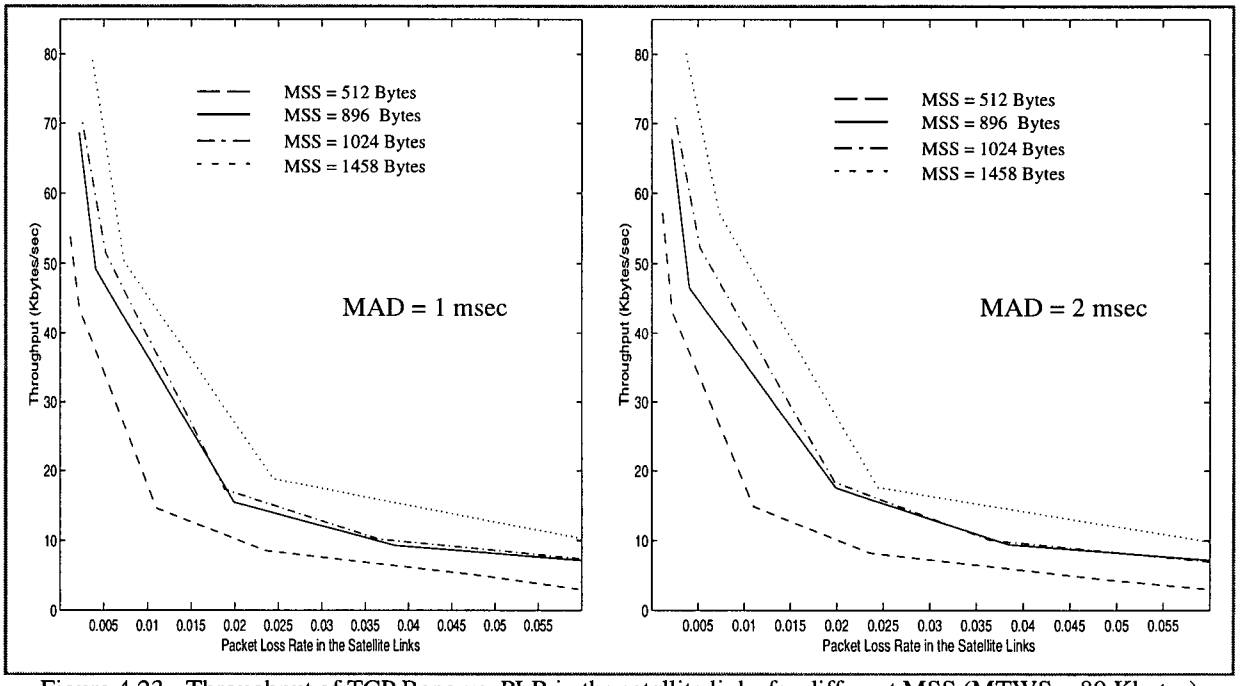

Figure 4.23 Throughput of TCP Reno vs. PLR in the satellite links for different MSS (MTWS = 80 Kbytes)

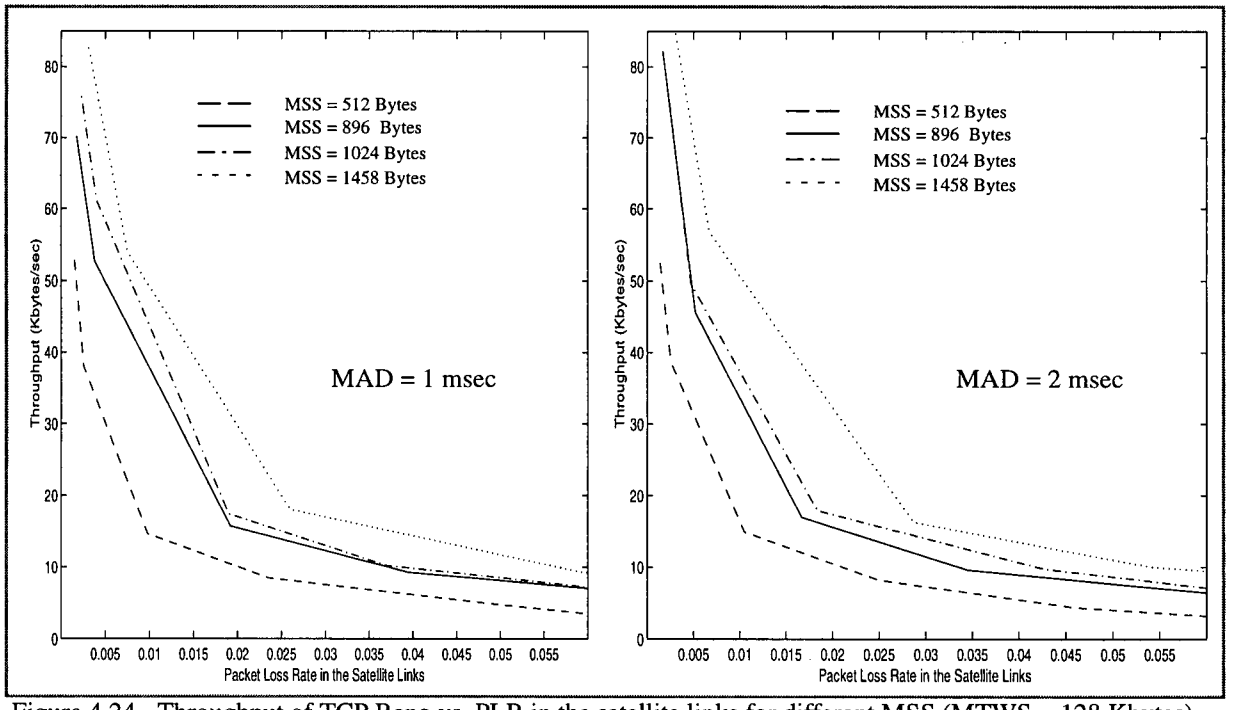

Figure 4.24 Throughput of TCP Reno vs. PLR in the satellite links for different MSS (MTWS = 128 Kbytes)

Figure 4.25 shows the throughput of TCP Reno with a MSS of 1458 bytes as a function of the PLR for different MTWSs and different MADs. At low PLRs (less than 0.03 in this case),

TCP Reno with a MAD of 2 msec has higher throughput than that with a shorter MAD for different MTWSs. However, TCP Reno with a MAD of 1 msec has slightly better throughput than that with a longer MAD for different MTWSs at a relatively high PLR in the satellite link.

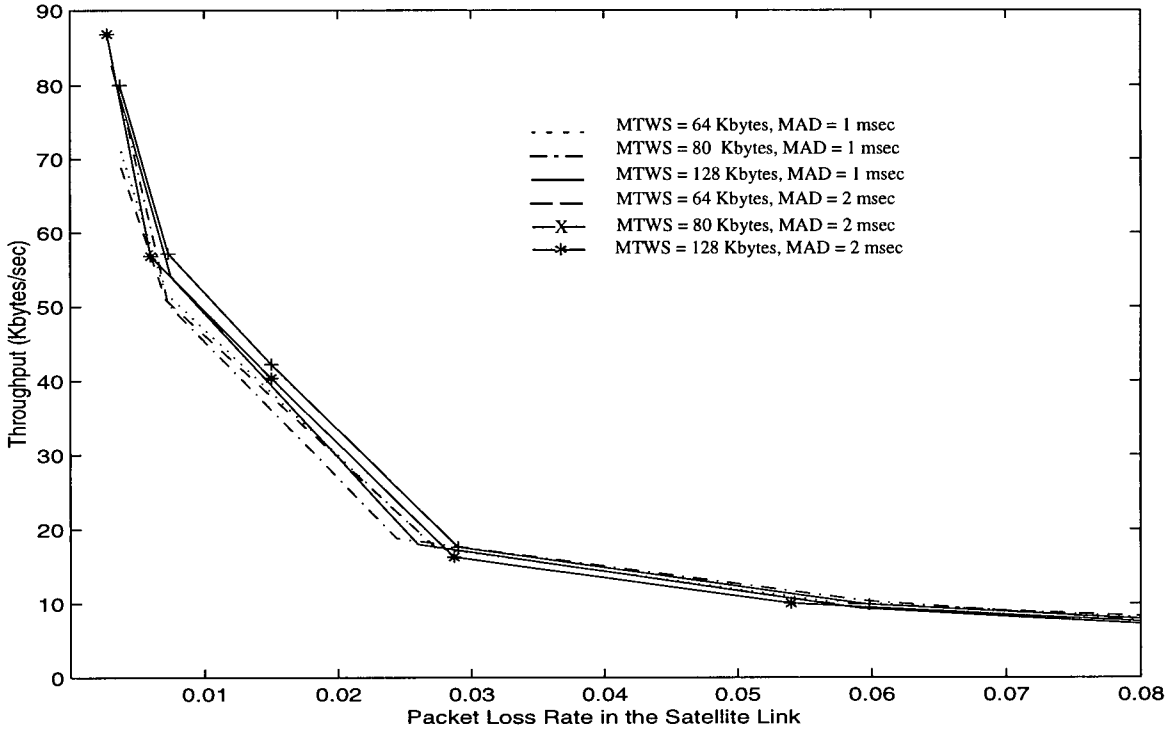

Figure 4.25 Throughput of TCP Reno with a segment size of 1458 bytes vs. the PLR in the satellite link

At a low PLR (< 0.03 in this case), TCP Reno with a MTWS of 80 Kbytes has much better throughput than that with other MTWSs. In the range of the PLR between 0.03 to 0.07, the throughput of TCP Reno with a MTWS of 80 Kbytes is slightly better than that with the other MTWSs. Beyond a PLR of 0.07, the throughput of TCP Reno with different MTWSs and different MAD is too close to be statistical significance. Figure 4.25 proves that the upper and lower bounds of the MTWS and the MAD do not have significant effects on the throughput performance of TCP Reno. In general, a MTWS of 80 Kbytes, a MAD of 2 msec, and a MSS of 1458 bytes give a reasonably high throughput for TCP Reno over a whole range of PLRs in the satellite channel for this HWW environment.

Figure 4.26-4.28 show the throughput of TCP Vegas as a function of the PLR in the satellite links for different MSSs under different TCP parameters settings. For different MADs, TCP Vegas with a 1458 byte MSS seems to have much better throughput then that with the other MTWSs (128 Kbytes and 80 Kbytes) but slightly better throughput than that with a 64 Kbyte MTWS. However, TCP Vegas with a 512 byte MSS has much lower throughput performance than that with the other MSSs. For different MADs, with , TCP Vegas with a 1458 byte MSS and a 128 Kbyte MTWS or a 80 Kbyte MTWS has roughly double better throughput than that with a 512 byte MSS, 1.5 times better throughput than that with a 896 byte segment size, and slightly better throughput that with a 1024 byte segment size. For different MADs, the MSS only has a slight effect on the throughput of TCP Vegas with a 64 Kbyte MTWS over the whole range of PLRs in the satellite channel.

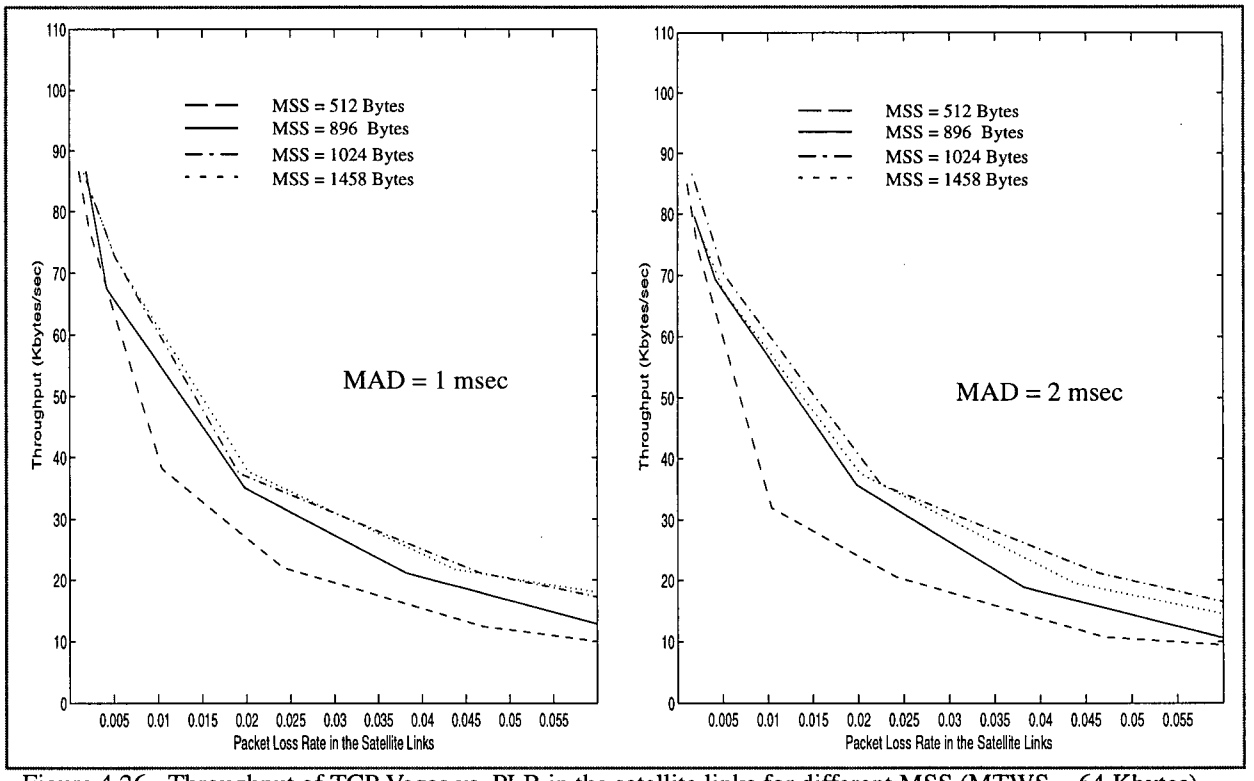

Figure 4.26 Throughput of TCP Vegas vs. PLR in the satellite links for different MSS (MTWS = 64 Kbytes)

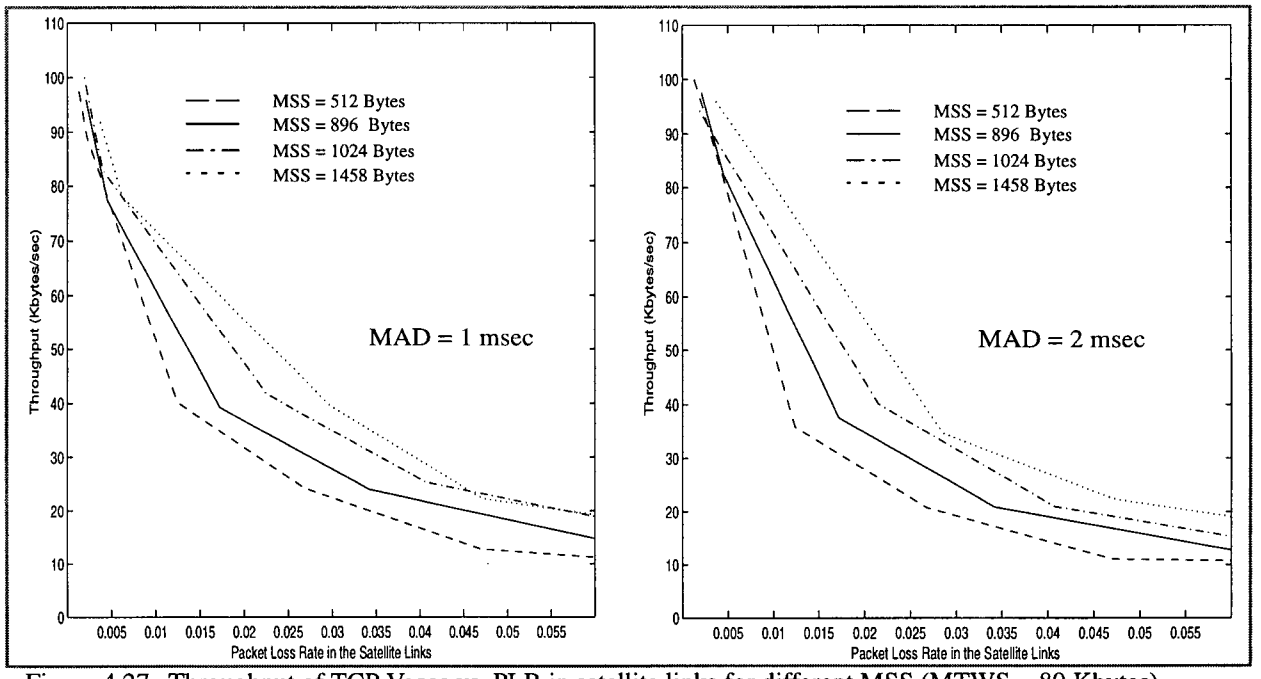

Figure 4.27 Throughput of TCP Vegas vs. PLR in satellite links for different MSS (MTWS = 80 Kbytes)

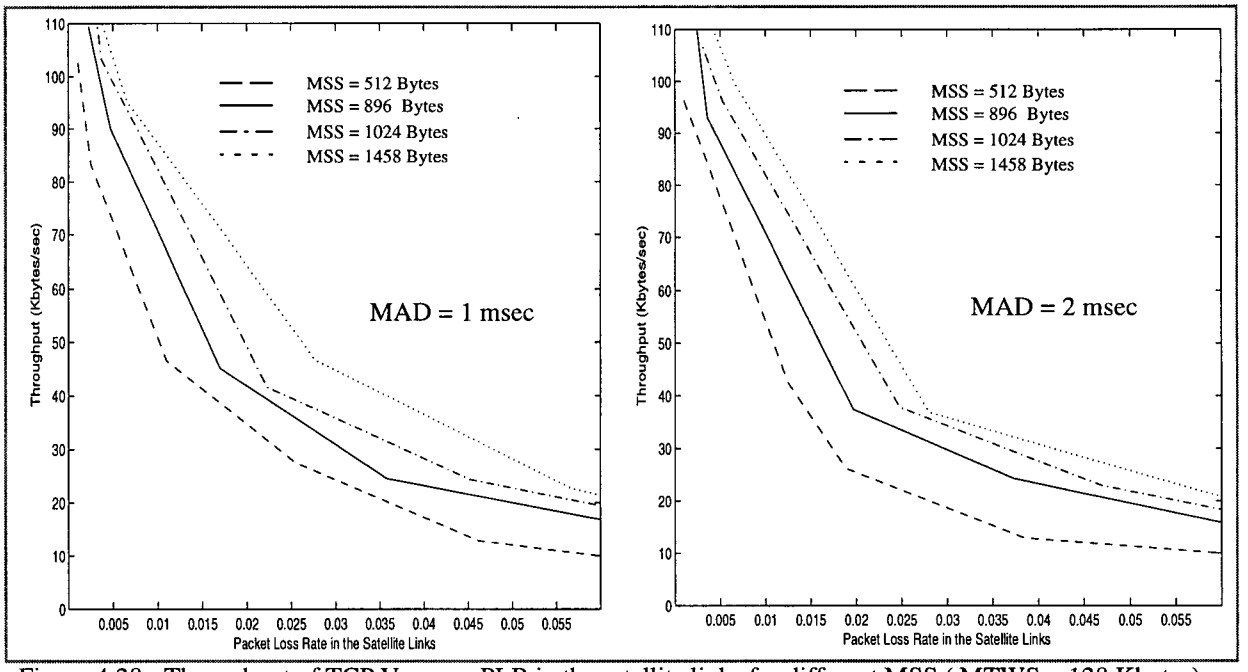

Figure 4.28 Throughput of TCP Vegas vs.PLR in the satellite links for different MSS (MTWS = 128 Kbytes)

Figure 4.29 shows the throughput of TCP Vegas with a 1458 bytes MSS as a function of the PLR in satellite link for different MTWSs and different MADs. The throughput of TCP Vegas

with a MAD of 10 msec, overall, is better than that with other MADs for different MTWSs over the whole range of the PLRs. At a low PLR (less than 0.07 in this case), TCP Vegas with a 128 Kbyte MTWS has much better throughput than that with other MTWSs, whereas TCP Vegas with a 64 Kbyte MTWS has much worse throughput than that with other MTWSs over the whole range of PLRs in the satellite channel. Beyond a PLR of 0.07, TCP Vegas with a 128 Kbyte MTWS and a MAD of 10 msec is still slightly better than that with other MTWSs

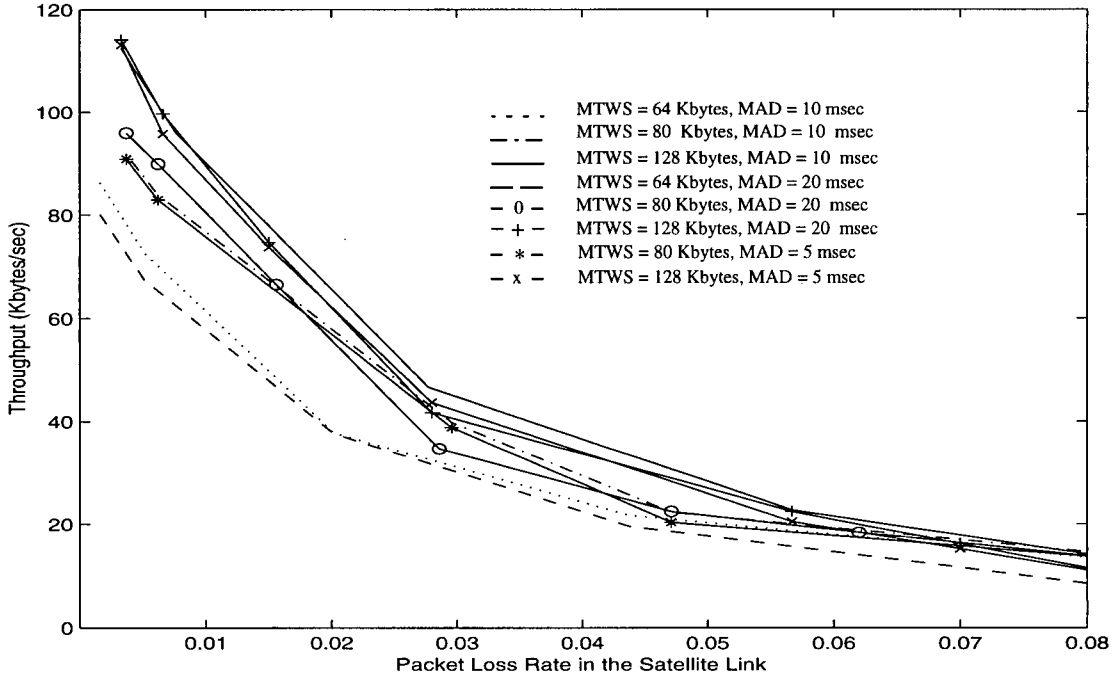

Figure 4.29 Throughput of TCP Reno with a segment size of 1458 bytes vs. the PLR in the satellite links

Compared with figure 4.25, figure 4.29 shows that the upper and lower bounds of the MTWS and the MAD have greater effect on throughput performance of TCP Vegas than on that of TCP Reno. In general, TCP Vegas with a 128 Kbyte MTWS, a 10 millsecond MAD, and a 1458 byte MSS, overall, has a reasonably high throughput performance over a whole range of PLRs in the satellite channel. Finally, the jointly optimal values of the chosen TCP parameters for TCP Vegas, TCP Reno and TCP RFC with a 0.3 second average ETE delay for different PLRs in the satellite channel are summarized in table 4.4.

Table 4.4 The suggested optimal TCP paramenters for different TCP implementations

TCP Parameters	TCP Vegas	TCP Reno	TCP RFC
MTWS	128 Kbytes	80 Kbytes	16 Kbytes
MAD	10 msec	2 msec	0.5 msec
MSS	1458 Bytes	1458 Bytes	1458 Bytes

The throughput performance of the TCP implementations with their optimal TCP parameters as a function of the PLR and the average ETE delay are presented in figure 4.30 and 4.31, respectively. Figure 4.30 shows that the throughput of all TCP implementations with their optimal TCP parameters is still decreased gradually when the PLR in the satellite channel is increased. The state model nearly has no effect on the throughput performance of TCP RFC, a slight effect on the throughput of TCP Reno, a great effect on the throughput of TCP Vegas. Compared with figure 4.3, figure 4.30 shows that the throughput performance of the different TCP implementations is significantly improved after jointly optimizing the TCP parameters for each TCP implementations over a whole range of PLRs. TCP Vegas with its optimal parameters has an approximately 34% to 100% better throughput than that without optiminzing its TCP parameters when the PLR varies from 0.001 to 0.06. TCP Reno with its optimal TCP parameters has an approximately 76% to 500% better throughput than that without optiminzing its TCP parameters when the PLR varies from 0.001 to 0.06. TCP RFC with its optimal TCP parameters, in general, has a approximately 300% better throughput than that without optiminzing its TCP parameters over the whole range of PLRs.

Table 4.5 shows the throughput comparison of TCP Vegas, TCP Reno and TCP RFC with their optimal parameters under a 0.3 second average ETE delay condition. From table 4.5, TCP Vegas has 23% better throughput than TCP Reno and more than 700% better throughput than TCP RFC under a clear sky condition. At a PLR of 0.06, TCP Vegas still has more than 200% better throughput than TCP Reno and around 500% better throughput than TCP RFC.

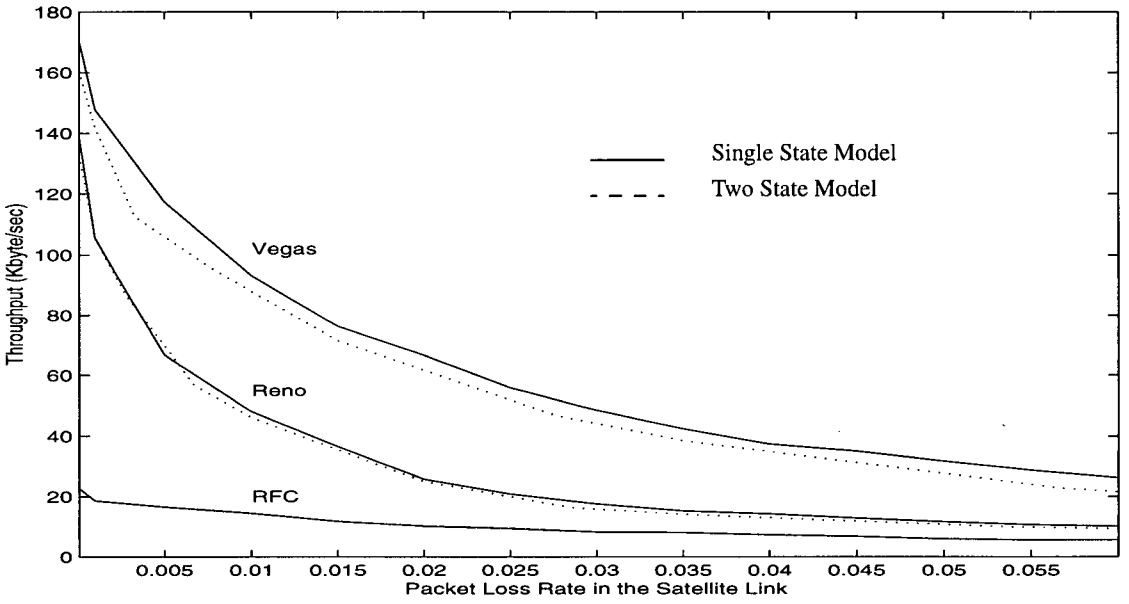

Figure 4.30 Throughput of different TCP implementations vs. the PLR in the satellite channel

Table 4.5 Throughput in Kbytes/sec of TCP versions with the ETE delay of 0.35 sec

	TCP Vegas	**TCP Reno**	**TCP RFC**
Throughput under a clear sky condition	170.31	138.56	23.06
Throughput ratio	7.386	6.008	1.0
Throughput at PLR of 0.02	63.61	25.66	10.31
Throughput ratio	6.170	2.489	1.0
Throughput at PLR of 0.04	37.42	14.31	7.3
Throughput ratio	5.126	1.960	1.0
Throughput at PLR of 0.06	26.16	10.26	5.65
Throughput ratio	4.630	1.816	1.0

Figure 4.31 shows the ETE delay has greater effect on TCP Vegas and TCP Reno than on TCP RFC; whereas the ETE delay has much less effects on the throughput of TCP RFC than on the others. When the ETE delay increases, the throughput of all TCP implememations is linearly decreased. Compared with figure 4.5, figure 4.31 show that TCP Reno with its optimal TCP parameters has significantly improvement in its throughput than the other TCP implementations.

Table 4.6 shows the throughput comparison of TCP Vegas, TCP Reno and TCP RFC with their optimal parameters at a 0.001 PLR in the satellite channel. After jointly optiminzing the key TCP parameters for the different TCP implementations, table 4.6 shows that TCP Vegas still has nearly 40% better throughput than TCP Reno and more than 700% better throughput than TCP RFC over the whole range of average ETE delays. However, all TCP implementations perform badly under a large ETE delay condition.

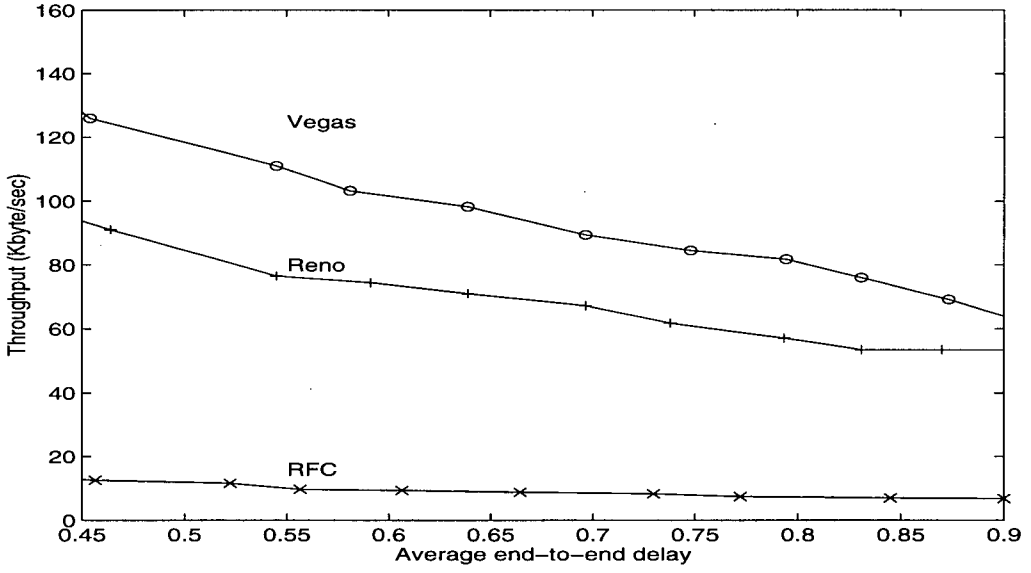

Figure 4.31 Throughput of different TCP implementations vs. the average ETE delay at a 0.001PLR

Table 4.6 Throughput in Kbytes/sec of different TCP versions at a PLR of 0.001

	TCP Vegas	**TCP Reno**	**TCP RFC**
Throughput at ete delay of 0.35	147.82	105.52	18.57
Throughput ratio	7.96	5.68	1.0
Throughput at ete delay of 0.65 sec	102.23	69.13	8.87
Throughput ratio	11.53	7.79	1.0
Throughput at ete delay of 0.85 sec	81.69	53.45	6.71
Throughput ratio	12.17	7.97	1.0

Chapter 5 Analysis of TCP in a Mobile Environment

In this chapter, the performance of two TCP implementations (TCP Vegas and TCP Reno) in an environment of mobile radio channels interworking with the Internet (MRCII) is studied and compared. Optimization of some key TCP parameters for the TCP implementations in this MRCII environment under different conditions is presented.

5.1 Cellular Mobile Communications

A schematic topology of a mobile computing environment consisting of wireless data networks interworking with the Internet is shown in Figure 5.1. The wide geographic area is divided into radio cells. Each cell has a base station, providing the radio interface with mobile hosts within its coverage area. Each base station is connected either to a switching office such as a mobile switching centre or directly to a gateway. Gateways may connect the remote mobile hosts via the Internet to fixed hosts attached to LANs.

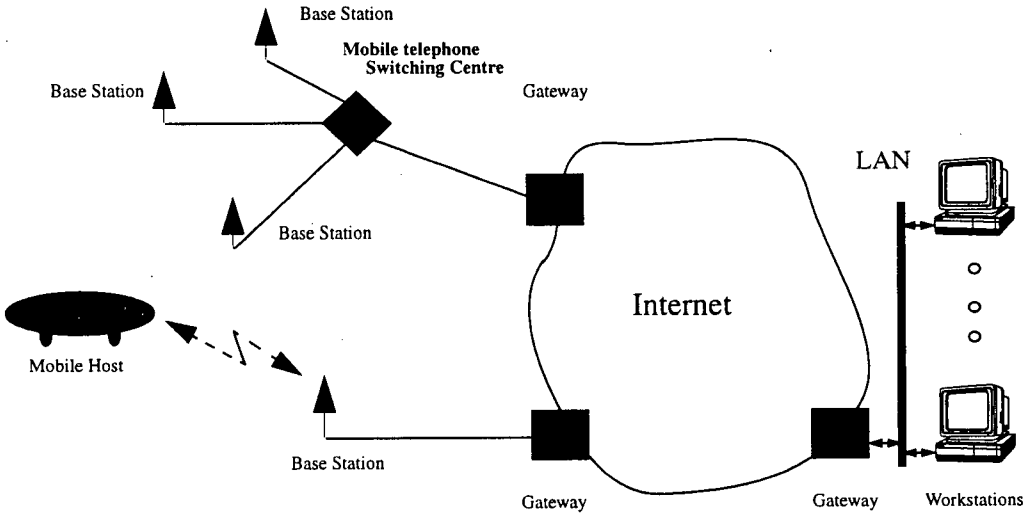

Figure 5.1 Interconnection between a mobile host and Internet fixed hosts

The mobile switching center has capability to separate the voice and data traffic, and data packets are processed and forwarded over the Internet. For example, in the CDPD system, this function is performed by an intermediate system called the Mobile Data Intermediate System. The TCP data segment transmission over the Internet is described in section 5.1. A gateway is used to connect the mobile data network with the Internet. When the gateway receives a packet from the Internet for a mobile host which it currently controls, it simply forwards the packet to an appropriate base station through which the mobile host can be reached. When expected packets are received in the mobile host, the corresponding acknowledgment packets are returned immediately from the destination mobile host to the source node.

From the simulation results in chapter 4, TCP RFC performs poorly in a high PLR environment. Therefore, only TCP Vegas and TCP Reno are considered in the following discussion. Because of the variable propagation changes environment and limited maximum data rate in mobile data networks, optimization of a number of TCP dependent parameters for both TCP implementations are necessary to improve TCP throughput. Aspects of communications related to the data link and the physical layer are beyond the scope of this thesis; therefore, no data link layer error recovery is assumed.

5.2 Simulation Model

Figure 5.2 shows the simulation model for a typical IP network incorporating with the mobile data networking environment. The simulation model includes a number of nodes and is divided into a mobile radio section and a Internet section and interconnected by a gateway. The mobile radio section models the characteristics of the mobile channel described in section 3.2. The Internet section models the characteristics of the Internet traffic described in section 3.3. In the simulation model, data packets are sent from a fixed source node interconnected with a LAN

to a destination mobile host, and the corresponding ACK packets are returned immediately from the destination mobile host to the source node. Passing through the Internet section and the mobile radio section, the data and ACK packets are delayed and may be lost due to congestion, transmission errors in the media, or handoffs.

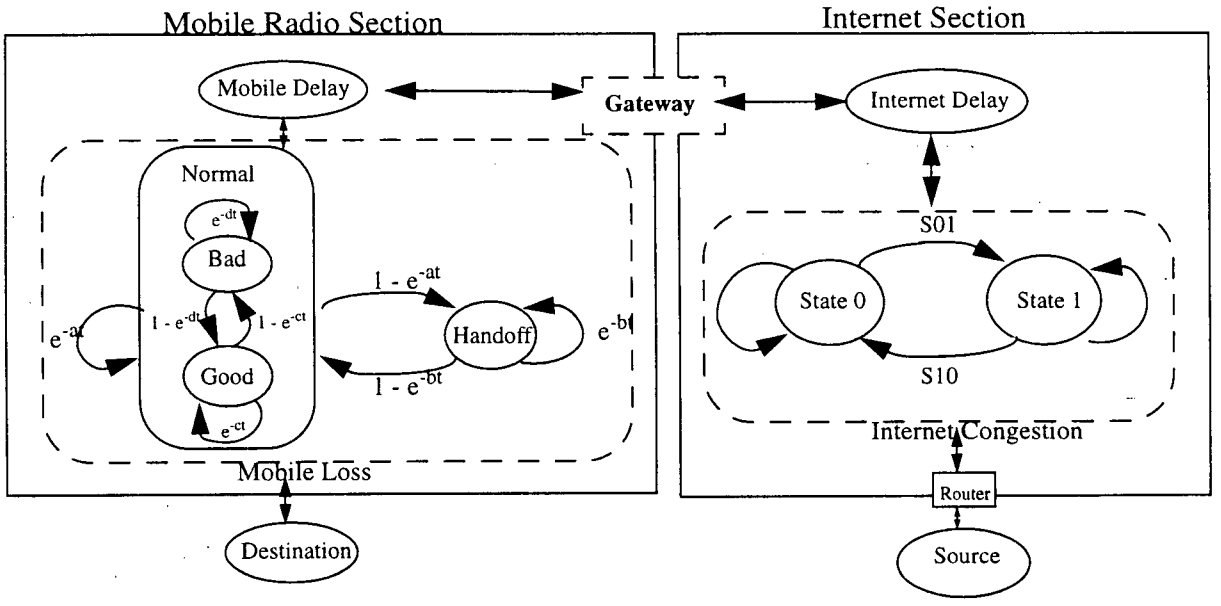

Figure 5.2 The mobile data network simulation model

5.3 Simulation Assumptions and Parameters

The simulation model is implemented in OPNET based on assumptions described in section 4.3. The fixed simulation parameters of the Internet section for all the simulations in this chapter are summarized in table 4.1. However, based on the simulation results of [7], the big window option need not be used in TCP connections with low bandwidth-delay products, as is the case here; therefore, the size of the TCP header is 20 bytes. The maximum window size in the receiver is assumed to be the same as that in the transmitter. The BER in the bad state within the normal state is much higher than that in the good state. The maximum data rate in the mobile data channel is assumed to be 19.2 Kbps. In order to maximize TCP performance over the mobile data

network interconnected with the Internet, the effects of the mobile speed, the PLR in the bad state within the normal state, the average time between handoffs (1/a), and the average time in performing handoff (1/b) on TCP throughput need to be analyzed. Moreover, key TCP parameters including the MTWS, the MAD, and the MSS need to be jointly optimized under different conditions.

5.4 Impact of Mobile Fading on TCP Throughput

This section presents the effects of the mobile speed and the PLR due to unreliable mobile radio links on the throughput performance of TCP Reno and TCP Vegas. Because of the signal-fading phenomenon, the average duration of fades (1/d) is inversely proportional to the mobile speed, and the STP from a good state to a bad state within the normal state ($1-e^{-ct}$) is directly proportional to the mobile speed (section 3.2), where 1/c is the average time between fades. When the mobile unit moves fast, the rate of the signal fluctuation is fast, but 1/d is short, thus causing a relatively high STP between the good state and the bad state even when there is no handoff involved. As a result, the loss probability for every transmitted packet is relatively high.

In this simulation, all the other key TCP parameters are fixed with values as suggested in [7][8], but the mobile speed and the PLR in the bad state within the normal state is varied. The mobile unit is assumed to be moving only within a cell so that no handoff process is involved. The MSS is assumed to be 512 bytes. The MTWS is assumed to be 8 Kbytes and the MAD is assumed to be 1 msec (typical use in Ethernet LANs). The RTT deviation coefficient is assumed to be 4. The thresholds (α and β) in TCP Vegas are assumed to be 1 and 3. Figure 5.3 shows the average PLR in the mobile radio channel as a function of the mobile speed for different PLRs in the bad state within the normal state. Figure 5.4 shows the throughput of TCP Reno and TCP Vegas as a function of the mobile speed for different PLRs in the bad state within the normal

state, respectively.

In figure 5.3, the average PLR in the mobile channel increases when the mobile speed is gradually increased because the STPs between the good state and bad state within the normal state are increased accordingly. In figure 5.4, the throughput of TCP Reno and TCP Vegas is gradually decreased when the mobile speed is increased for different PLRs in the bad state. By increasing the mobile speed, the rate of signal fluctuation is increased, thus causing a relatively high STP from the good state to the bad state within the normal state. As a result, the PLR over the mobile channel is relatively high even with a shorter 1/d. The mobile speed seems to have greater effects on the throughput of TCP Reno than on that of TCP Vegas. Over the whole range of mobile speeds, TCP Vegas seems to have slightly better throughput than TCP Reno for different PLRs in the bad state within the normal state.

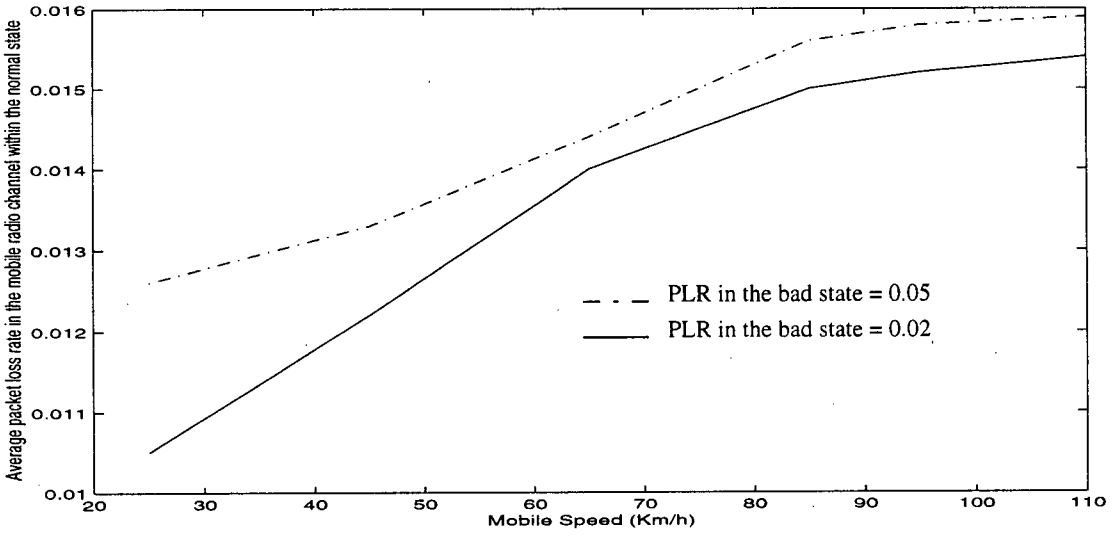

Figure 5.3 Average PLR in the mobile radio channel vs. mobile speed

Figure 5.5 shows a more detailed picture of the effects of the PLR in the bad state within the normal state on the throughput of TCP Reno and TCP Vegas at different mobile speeds. At a mobile speed of 45 km/h, TCP Vegas has approximately 0.5% to 4% better throughput than TCP

Reno when the PLR in the bad state varies from 0.001 to 0.35. However, at a mobile speed of 75 km/h, TCP Vegas has approximately 0.7% to 10% better throughput than TCP Reno when the PLR in the bad state varies from 0.001 to 0.35. Therefore, while too low a mobile speed may increase the 1/d but decrease the STPs between the two states, and too high a mobile speed may increase the rate of signal fluctuation. From figure 5.5, TCP Vegas seems to have a better response to the increased rate of signal fluctuation because of its faster error recovery and better congestion avoidance mechanisms.

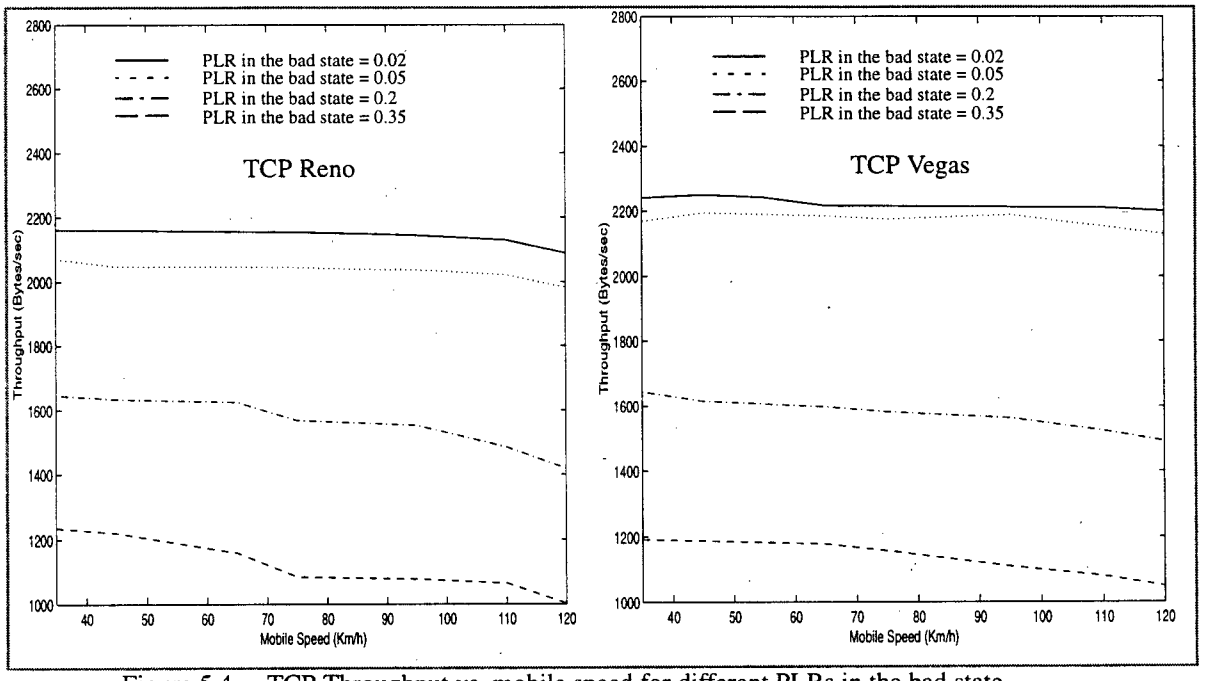

Figure 5.4 TCP Throughput vs. mobile speed for different PLRs in the bad state

Figure 5.6 shows the congestion window response in both TCP Reno and TCP Vegas as a function time for different mobile speeds at a PLR of 0.05 in the bad state within the normal state. When the mobile unit moves slowly in a relatively high PLR environment, 1/d is relatively long, thus causing several packets to be corrupted per each full transmit window size. As a result, not only the slow-start mechanism but also the exponential back-off algorithm are relatively active, causing unnecessarily long communication pauses. When the mobile unit moves fast in a

relatively high fading environment, the rate of the congestion window fluctuation is increased, thus the congestion window does not have enough time to expand before the next mobile radio channel interruption occurs.

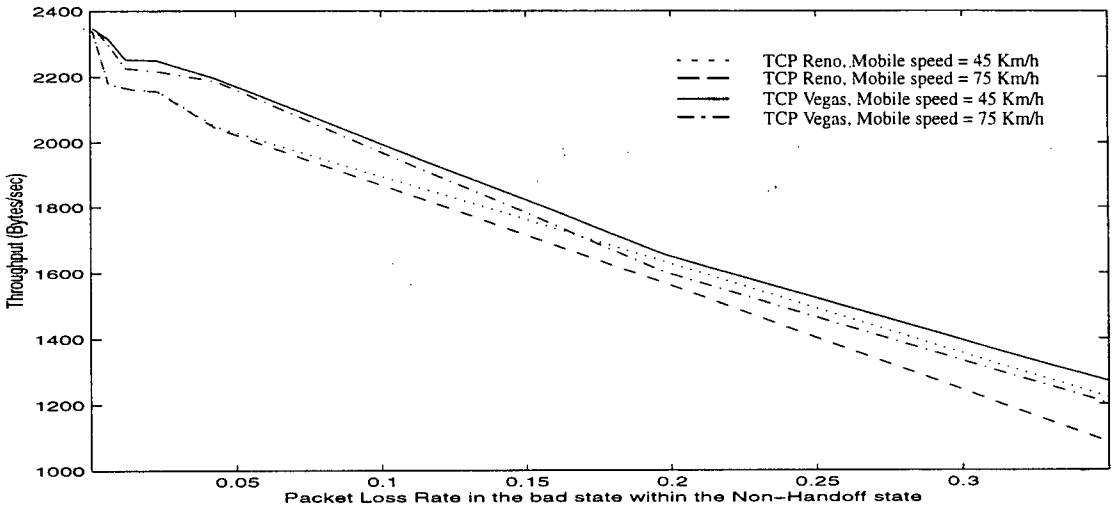

Figure 5.5 TCP throughput vs. PLR in the bad state for different mobile speeds

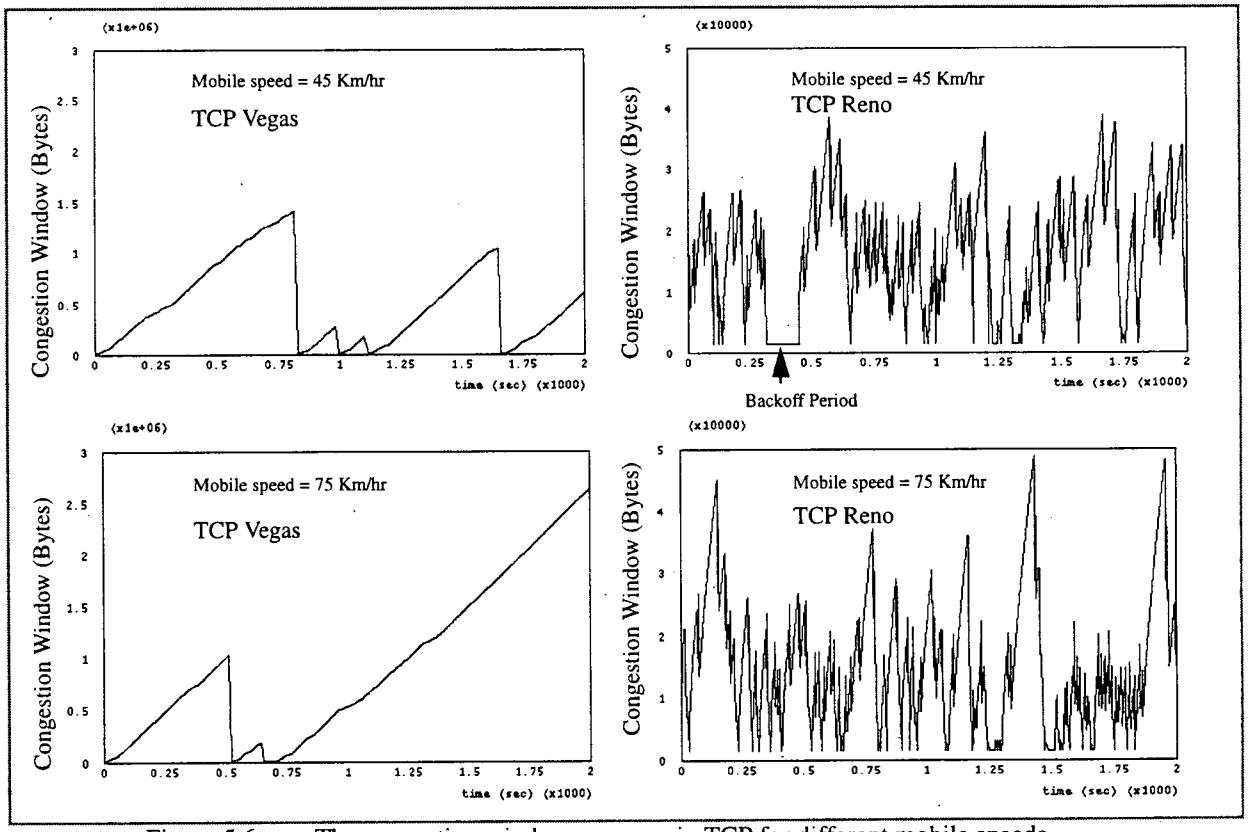

Figure 5.6 The congestion window response in TCP for different mobile speeds

Because of the longer 1/d, the exponential back-off reaction in TCP Reno at a mobile speed of 45 km/hr seems more active (see backoff period indicated with an arrow in figure 5.6) than that at 75 Km/h, degrading TCP throughput. At 45 Km/h, TCP Vegas has less frequent drops (reset to one MSS) of the congestion window than that at 75 Km/h. Because TCP Vegas has a better congestion avoidance and a faster error recovery mechanism, the rate of the congestion window fluctuation in TCP Vegas is much less than that in TCP Reno. The size of the congestion window in TCP Reno is limited by not only mobile radio channel noise but also congestion.

5.5 Impact of Handoff on TCP Throughput

This section presents the effects of the handoffs due to cell crossings on the throughput performance of TCP Reno and TCP Vegas. The 1/a is directly proportional to the size of the radio cells and inversely proportional to the mobile speed. The 1/b is directly proportional to the area of the handoff regions and inversely proportional to the mobile speed. All the TCP parameters are fixed to the same values as in section 5.4 but only the 1/a and the 1/b are varied by varying the radius of the radio cells (from 0.5 Km to 5 Km) and the handoff regions for different mobile speeds. For every 1/a sec on average, when the mobile host moves from one cell to another cell, the wireless channel is assumed to be unavailable (in the handoff state) for an average of 1/b sec. An average PLR in the bad state within the normal state is assumed to be 0.05.

Figures 5.7, 5.8, and 5.9 show the throughput performance of TCP Reno and TCP Vegas as a function of 1/a for a mobile speed of 45 Km/h, 75 Km/h, and 120 Km/h, respectively, under different 1/b values, where the handoff region = 50 m, 100 m and 200 m are considered. They show that 1/a and 1/b have a great influence on the throughput of both TCP Reno and TCP Vegas for different mobile speeds. As expected, the throughput of both TCP implementations is decreased as 1/b is increased and increased as 1/a is increased at different mobile speeds. A

shorter 1/a at faster mobility rate causes a larger degradation of the throughput of both TCP implementations. A longer handoff delay has a higher PLR in the mobile radio link. At different mobile speeds, TCP Vegas still has better throughput than TCP Reno over the whole range of 1/a for different 1/b values. For different handoff regions and different sizes of the radio cells, the throughput of TCP Reno and TCP Vegas decreases approximately 2% when the mobile speed is increased from 45 Km/h to 75 Km/h and approximately 6% when the mobile speed is increased from 45 Km/h to 120 Km/h.

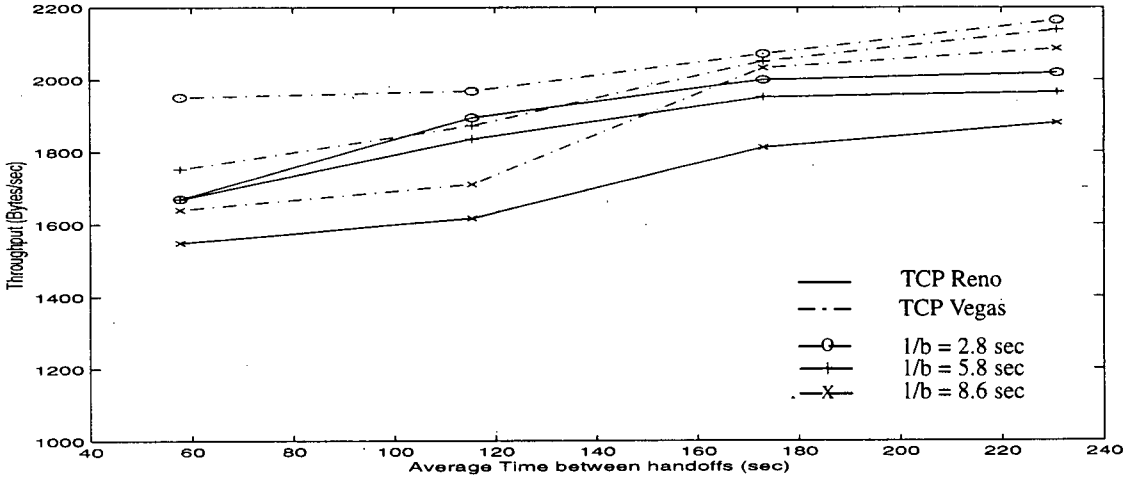

Figure 5.7 TCP throughput vs 1/a for different 1/b values at a mobile speed of 45 Km/h

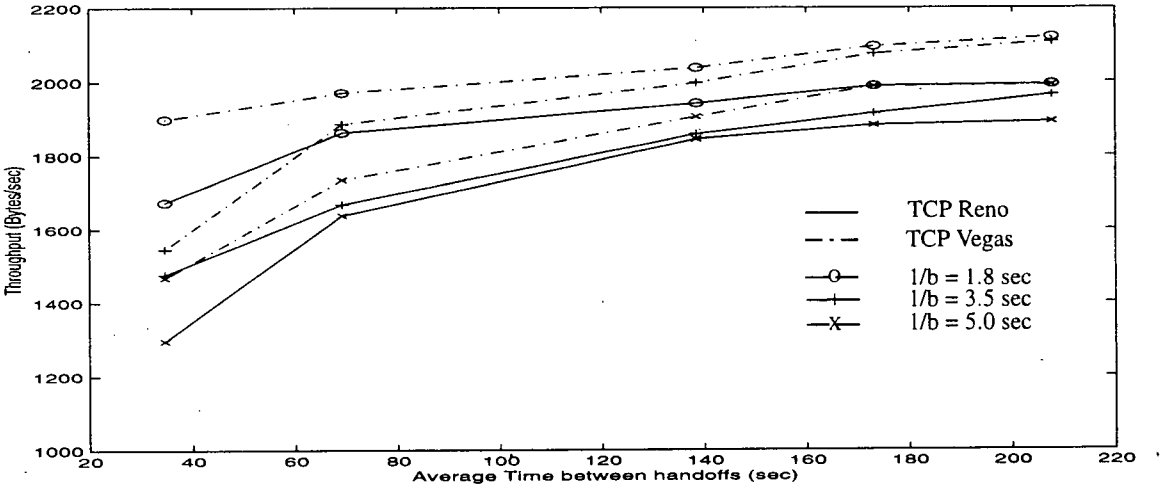

Figure 5.8 TCP throughput vs 1/a for different 1/b values at a mobile speed of 75 Km/h

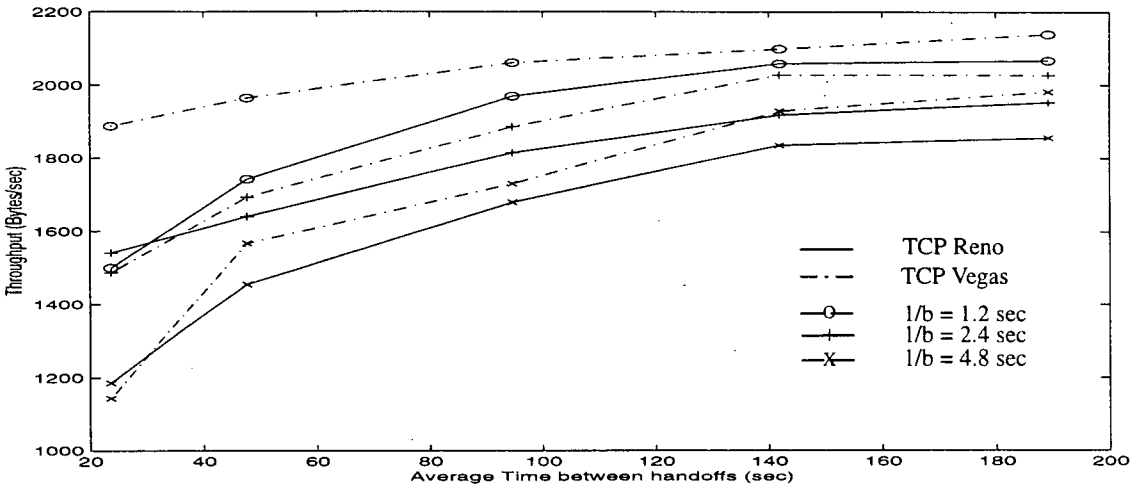

Figure 5.9 TCP throughput vs 1/a for different 1/b values at a mobile speed of 120 Km/h

Figure 5.10 shows a detailed graph of the effects of 1/a on the throughput performance of TCP Reno and TCP Vegas as a function of 1/a for different 1/b values at a mobile speed of 75 Km/h. It shows that the throughput of both TCP implementations is seriously affected by 1/a. As 1/a is reduced, for different 1/b values, the throughput of both TCP implementations decreases because there are more and more retransmissions. When 1/a is further reduced, the recovery of the TCP connection from a channel interruption is not fast enough to expand the current congestion window size before the next channel interruption occurs. These channel interruptions violate most of the conditions required for reliable TCP operation. As a result, an incorrect TCP time-out for every segment is estimated and causes the serious degradation of TCP throughput.

However, for a larger 1/a, the duration of the channel availability is sufficiently long to open up the congestion window size before the next interruption occurs. Therefore, there is sufficient bandwidth not only to retransmit all the lost segments but also to send more segments into the network. TCP Vegas, overall, has approximately 25% to 14% better throughput than TCP Reno for a smaller 1/b and has approximately 12% to 5% better throughput than TCP Reno for a larger 1/b when 1/a varies from 35 sec to 180 sec.

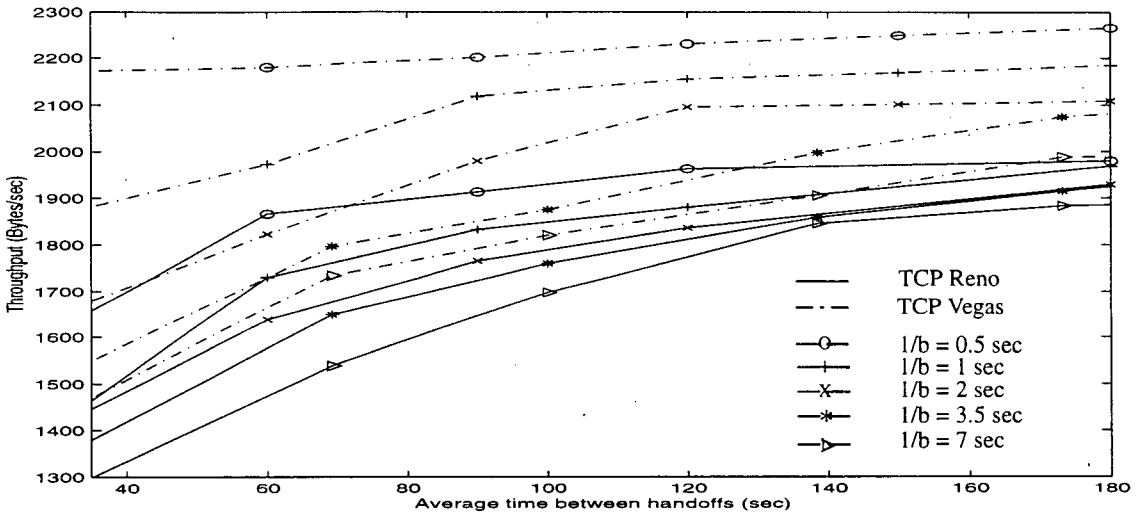

Figure 5.10 TCP throughput vs. 1/a for different 1/b values at mobile speed of 75 Km/h

Figure 5.11 plots the sequence numbers of the received TCP segments as a function of time for 1/b = 3.5 sec and 1/a = 70 sec at a mobile speed of 75 Km/h to present a more detailed graph of the behavior of the end-to-end connection for TCP Vegas and TCP Reno. TCP Vegas maintains a higher and more consistent throughput than TCP Reno. Because TCP Reno unnecessarily invokes the congestion control procedures more often than TCP Vegas for the duration of the connection, the extensive communication pauses significantly degrade TCP throughput. This phenomenon appears as the flat and empty regions of the curve.

Figure 5.12 plots the congestion window response in TCP Reno and TCP Vegas as a function of time for 1/b = 3.5 sec with different 1/a values at a mobile speed of 75 Km/h to present the behavior of the congestion window response in both TCP implementations for different 1/a values. TCP Reno, as expect, has more frequent drops of the congestion window than TCP Vegas for different 1/a values. When 1/a = 70 sec, both TCP implementations invoke the slow-start mechanism more often and have longer communication pauses than both TCP implementations when 1/a = 200 sec, thus degrading the TCP throughput performances. The flat and empty regions of the curve represent the communication pauses.

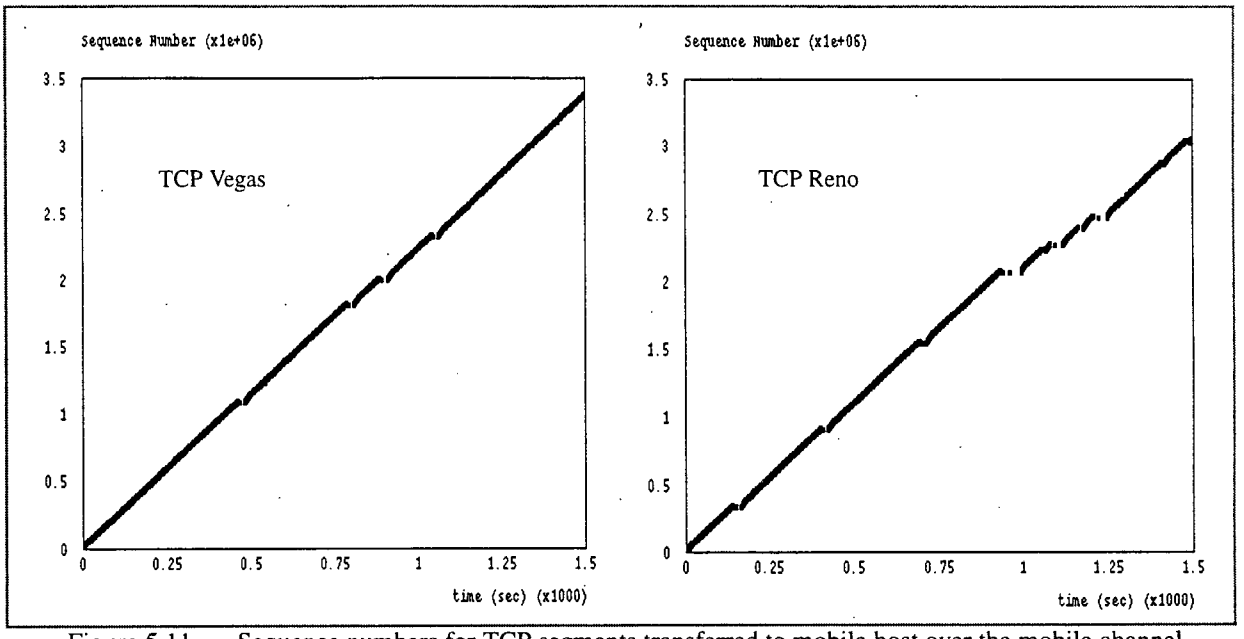

Figure 5.11 Sequence numbers for TCP segments transferred to mobile host over the mobile channel

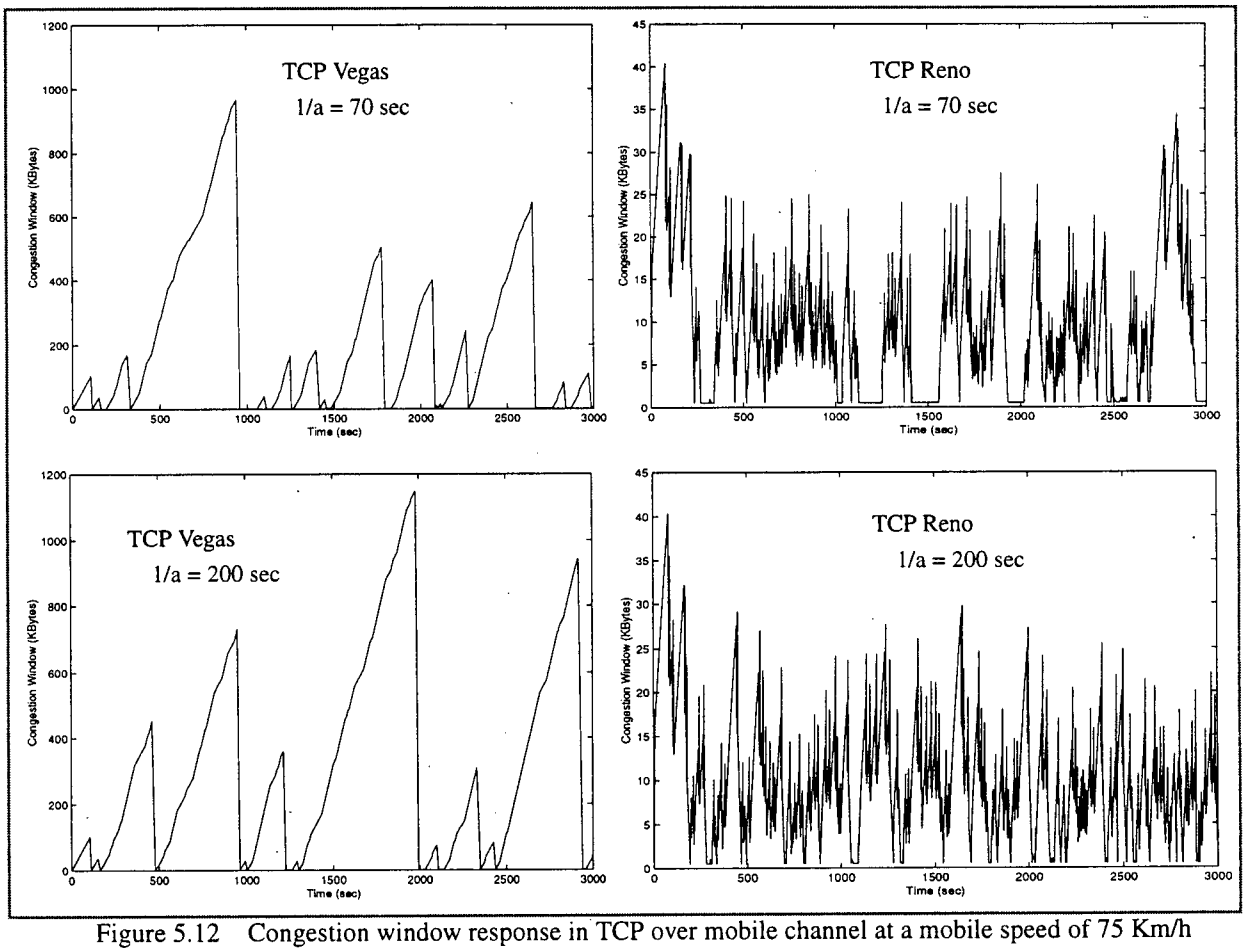

Figure 5.12 Congestion window response in TCP over mobile channel at a mobile speed of 75 Km/h

5.6 Impact of TCP Parameters on TCP Throughput

This section analyzes the effects of some key TCP parameters including the MTWS, the MAD, and the MSS on the throughput performance of both TCP Reno and TCP Vegas over mobile radio channels with severe fading and frequent handoffs, internetworking with the Internet. In this section, the thresholds (α and β) in TCP Vegas are assumed to be 1 and 3 [26], and the RTT deviation coefficient is assumed to be 4 [17].

5.6.1 Effects of Maximum Transmit Window Size

Many pervious studies have showed that the MTWS is a key TCP parameter which has a strong influence on TCP throughput in the mobile fading environment [6][7][8][10]. In this subsection, the effects of the MTWS on the throughput performance of both TCP implementations are analyzed and compared. The optimal MTWSs for both TCP implementations to maximize throughput under different conditions are presented. All other network components and TCP parameters are fixed for performance evaluation by only varying the MTWS. The MSS is assumed to be 512 bytes as suggested in [7]. The MAD is assumed to be 1 msec. In this subsection, two scenarios are considered.

In the first scenario, the radius of the radio cells and the area of the handoff region are fixed at 2 Km (a lower bound of radio marco cells) and 100 m, respectively, but the speed of the mobile unit is varied. In the second scenario, the mobile speed is fixed at a 75 Km/h, but 1/a and 1/b are varied. The PLR in the bad state within the normal state is assumed to be 0.02 for both scenarios. Figure 5.13 shows the throughput of both TCP implementations in the first scenario as a function of the MTWS for different mobile speeds. Figure 5.14 and figure 5.15 show the throughput of both TCP implementations in the second scenario as a function of the MTWS for different 1/a and 1/b values, respectively.

Figure 5.13 shows that the MTWS seriously affects the throughput of both TCP implementations at different mobile speeds. The throughput of both TCP implementations is increased when the MTWS is increased. Beyond a certain MTWS, their throughput gradually decreases. At different mobile speeds, both TCP implementations with a smaller MTWS (< 8 Kbytes) have better throughput than that with a larger MTWS. However, too small a MTWS has greater effects on the throughput of TCP Vegas than on that of TCP Reno especially when the mobile unit moves significantly fast. Too small a MTWS limits TCP sending more data into the network when the channel staying in the normal state, thus seriously degrading TCP throughput. At a fast relatively mobile speed, both TCP implementations prefer to use a smaller MTWS because of the increasing rate of signal fluctuation. On the other hand, they prefer to use a larger MTWS at a relatively slow mobile speed in order to send more data segments when the channel staying in the normal state. At different mobile speeds, TCP Reno prefers to use a smaller MTWS than TCP Vegas because of the limitation of its congestion avoidance mechanism.

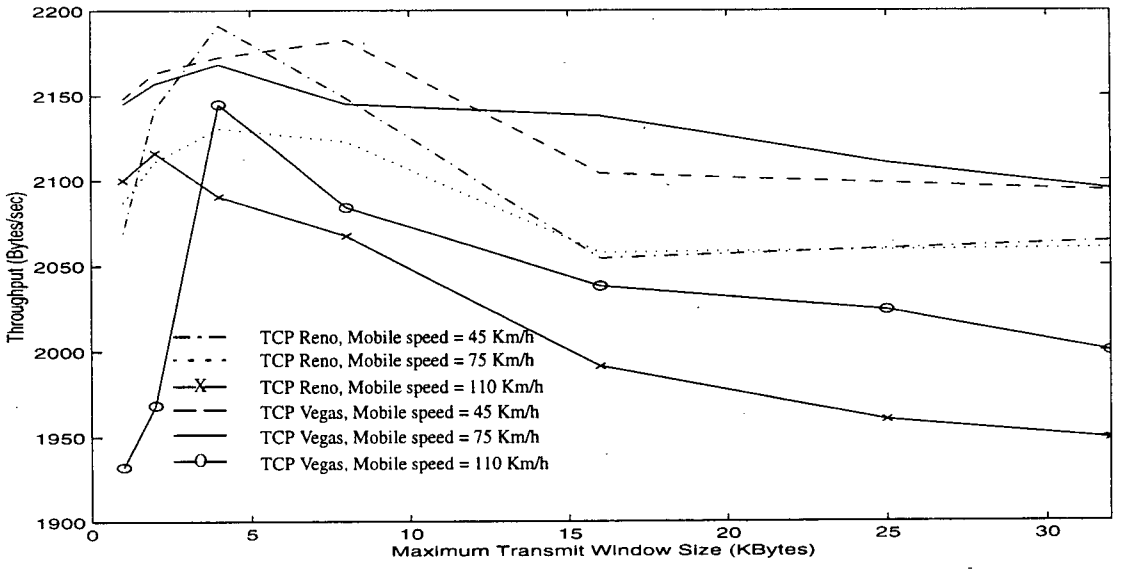

Figure 5.13 TCP throughput vs. MTWS for different mobile speeds

Figures 5.14 and 5.15 show that the MTWS has great effects on the throughput of both

TCP implementations for different 1/a and 1/b values. Figure 5.14 shows that both TCP implementations prefer to use a bigger MTWS for a longer 1/a and a smaller transmit window size for a shorter 1/a. It is because a bigger MTWS allows more data packet sending into the network during the time between handoffs. However, too large a MTWS requires much longer recovery time from the channel interruptions for the source host to go back to the nominal throughput level. With a bigger MTWS, both TCP implementations have difficulty recovering from multiple losses within each full sending window by using their error recovery mechanisms; therefore, most of the lost packets are recovered by the coarse-grain RTO mechanism. A smaller MTWS seems to give much better TCP throughput performance in this MRCII environment.

Figure 5.15 shows that both TCP implementations prefer to use a smaller MTWS for both 1/b considered values since a bigger transmit window size creates more packet losses during the time performing handoffs. Due to their congestion avoidance mechanisms, a bigger MTWS causes a long communication pause. It shows that a 4 Kbyte MTWS seems to give much better throughput for both TCP implementations for different 1/b values at a mobile speed of 75 Km/h.

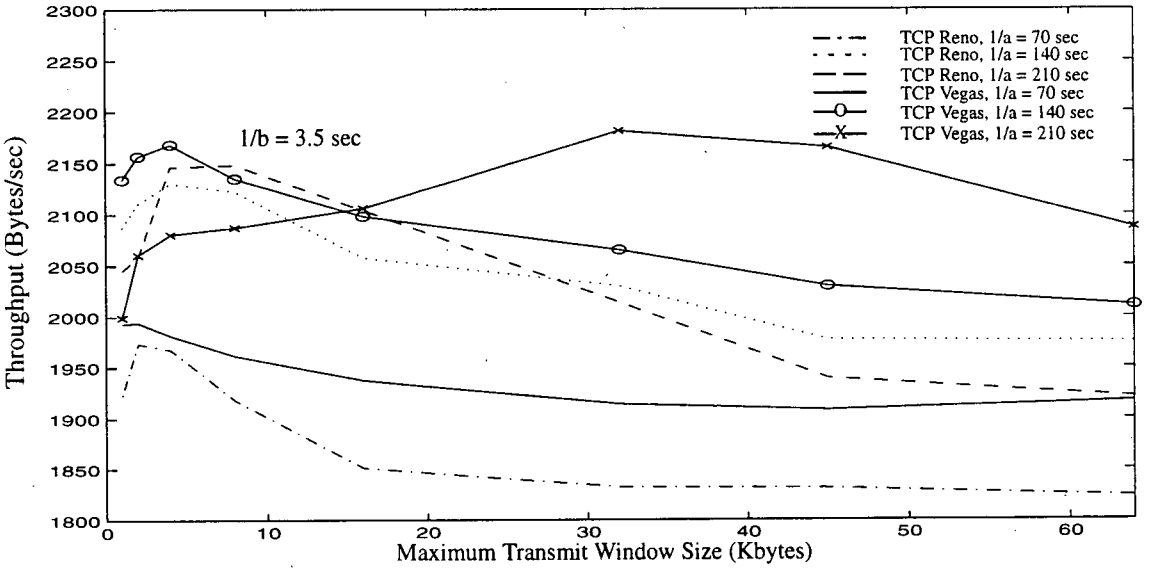

Figure 5.14 TCP throughput vs MTWS for different 1/a values

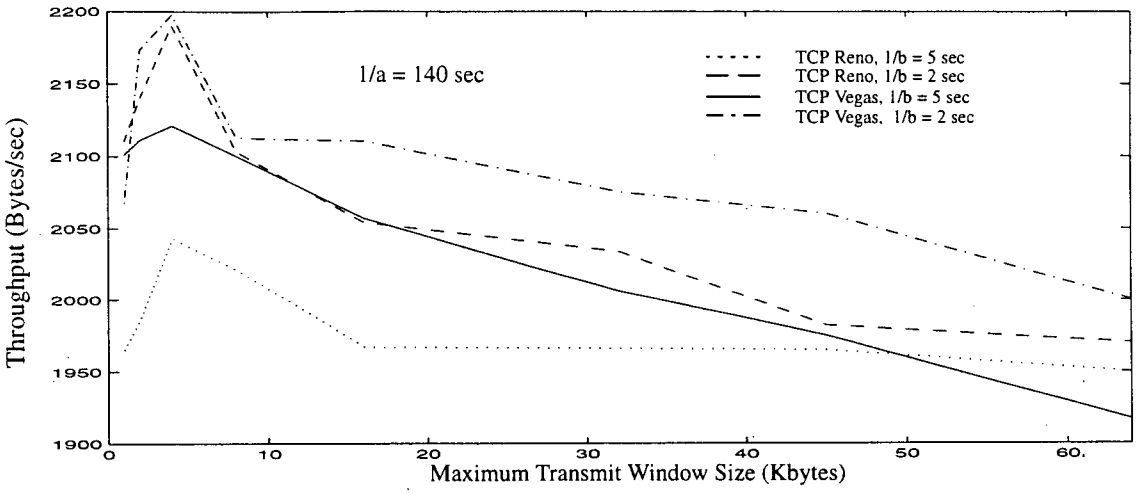

Figure 5.15 TCP throughput vs. MTWS for different 1/b values

Figure 5.16 illustrates the throughput comparison of TCP Reno and TCP Vegas, respectively, as a function of the MTWS for different PLRs in the bad state within the normal state with 1/a = 140 sec and 1/b = 2 sec at a mobile speed of 65 Km/h. The MTWS has great effects on the throughput of both TCP implementations at different PLRs. At a low PLR in the bad state within the normal state, their throughput increases as the MTWS increases. Beyond a certain MTWS, their throughput gradually decreases because a bigger MTWS creates more packet losses by overloading the network buffers. At a high PLR, their throughput sharply decreases when the MTWS increases. Because a larger MTWS requires a much longer recovery time from the channel interruptions, a smaller MTWS gives better TCP throughput.

Table 5.1 summarizes the optimal MTWSs for both TCP implementations under different conditions and their corresponding throughput. From table 5.1, it seems that a bigger MTWS is more suitable for TCP Reno and TCP Vegas at a relatively low mobility rate, but a smaller MTWS is more suitable for them at a relatively high mobility rate. For a longer 1/b and a high PLR, a small MTWS can improve both TCP throughput performances because of the relatively low PLR per each full sending window size. With a better congestion avoidance mechanism and a faster

retransmission mechanism, TCP Vegas, overall, has much better throughput than TCP Reno under most conditions.

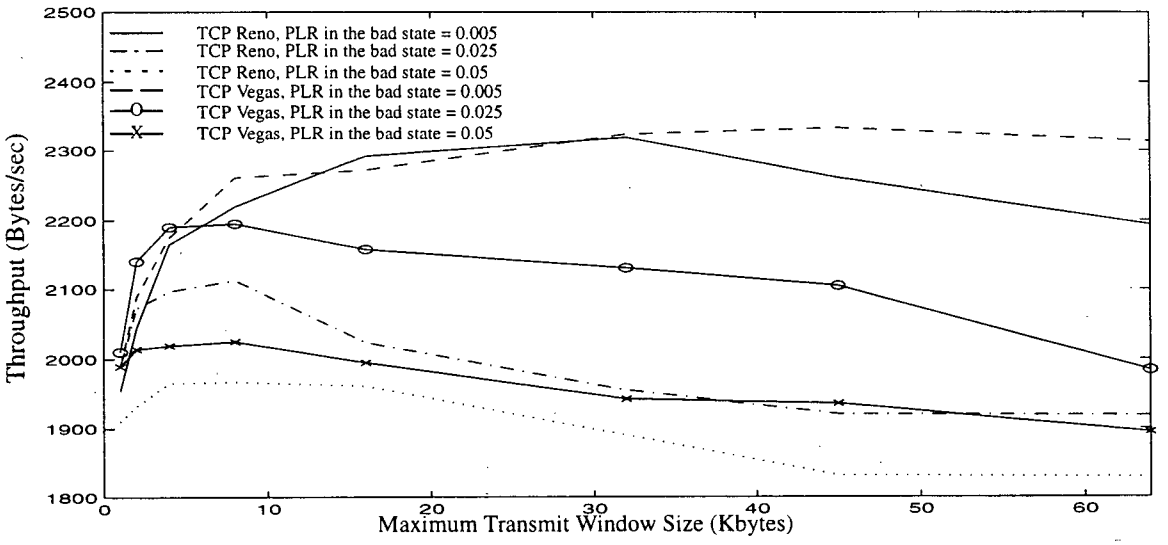

Figure 5.16 TCP throughput vs. MTWS for 1/a = 140 sec

Table 5.1 The optimal transmit window sizes for both TCPs under different conditions

	TCP Reno	TCP Vegas
Tx window size at mobile speed = 45 Km/h	4 Kbytes	8 Kbytes
Throughput (Bytes/sec)	2191	2182
Tx window size at mobile speed = 75 Km/h	4 Kbytes	4 Kbytes
Throughput (Bytes/sec)	2130	2168
Tx window size at mobile speed = 110 Km/h	2 Kbytes	4 Kbytes
Throughput (Bytes/sec)	2116	2144
Tx window size at 1/a = 70 sec	2 Kbytes	2 Kbytes
Throughput (Bytes/sec)	1973	1994
Tx window size at 1/a = 140 sec	4 Kbytes	4 Kbytes
Throughput (Bytes/sec)	2130	2168
Tx window size at 1/a = 210 sec	8 Kbytes	32 Kbytes
Throughput (Bytes/sec)	2148	2181

5.6.2 Effects of Maximum ACK Delay

This subsection mainly studies the effects of the MAD on TCP throughput performance in BRCII environment. All the other parameters are fixed by varying the MAD for different mobile speeds and different MTWSs. Between handoffs, an average PLR of 0.05 in the bad state within the normal state is assumed. The MSS is assumed to be 512 bytes. The radius of the radio cells and the area of the handoff region are fixed at 2 Km and 100 m, respectively. In the first scenario, the MTWS is assumed to be 16 Kbytes, but the mobile speed is varied. In the second scenario, the mobile speed is fixed at 75 Km/h, but the MTWS is varied. Figure 5.17 shows the throughput of TCP Reno and TCP Vegas as a function of the MAD at different mobile speeds. Figure 5.18 and 5.19 show that the throughput of TCP Reno and TCP Vegas as a function of the MAD for different MTWSs.

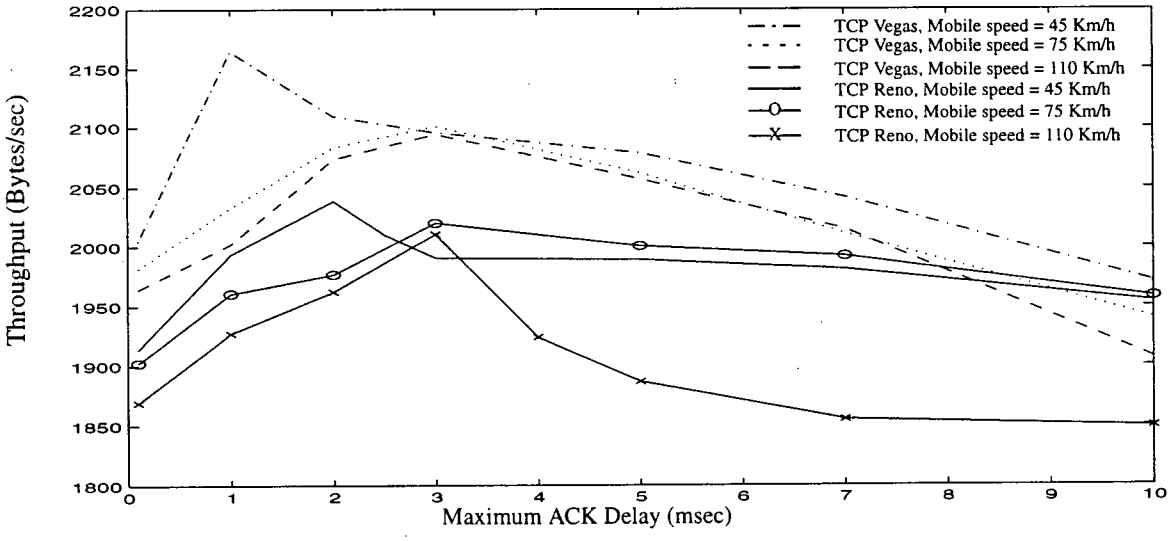

Figure 5.17 TCP throughput vs MAD at different mobile speeds

Figure 5.17 shows that the MAD greatly affects the throughput of both TCP Reno and TCP Vegas at different mobile speeds. Their throughput increases as the MAD increases; beyond a certain MAD, their throughput gradually decreases. At a relatively high mobile speed, it seems

that a longer MAD gives better throughput for both TCP implementations; On the other hand, a shorter MAD gives better throughput of that at a relatively slow mobile speed. It is because a longer MAD reduces the data transmission rate at the sender, particular in a high mobility environment, in order to reduce PLR during handoffs. At a lower mobile speed (e.g. 45 Km/h), a shorter MAD not only can increase the data transmission rate at the sender but also have fast error recovery response, thus improving the throughput of both TCP Reno and TCP Vegas.

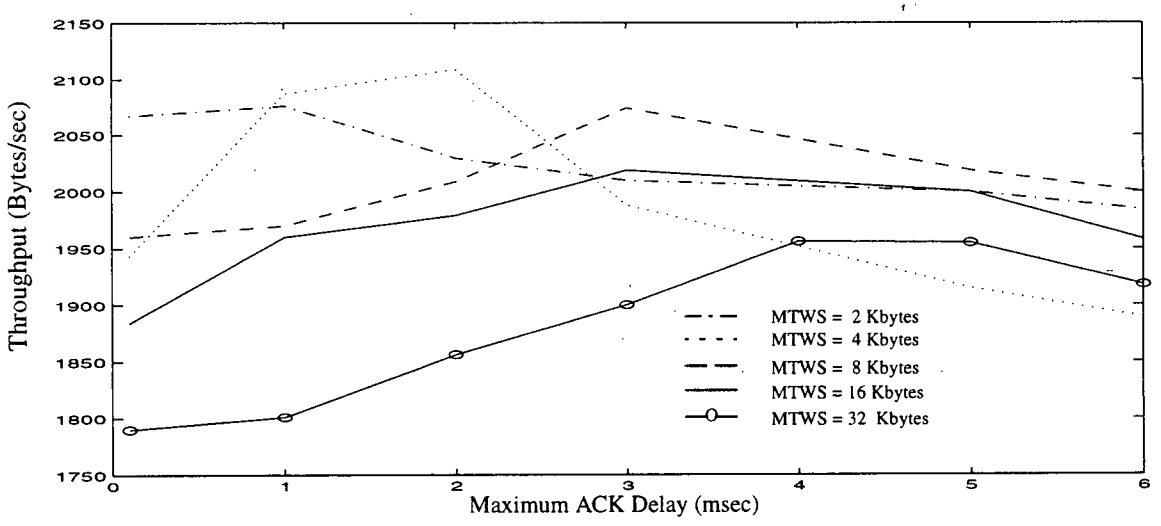

Figure 5.18 Throughput of TCP Reno vs. MAD for different MTWSs

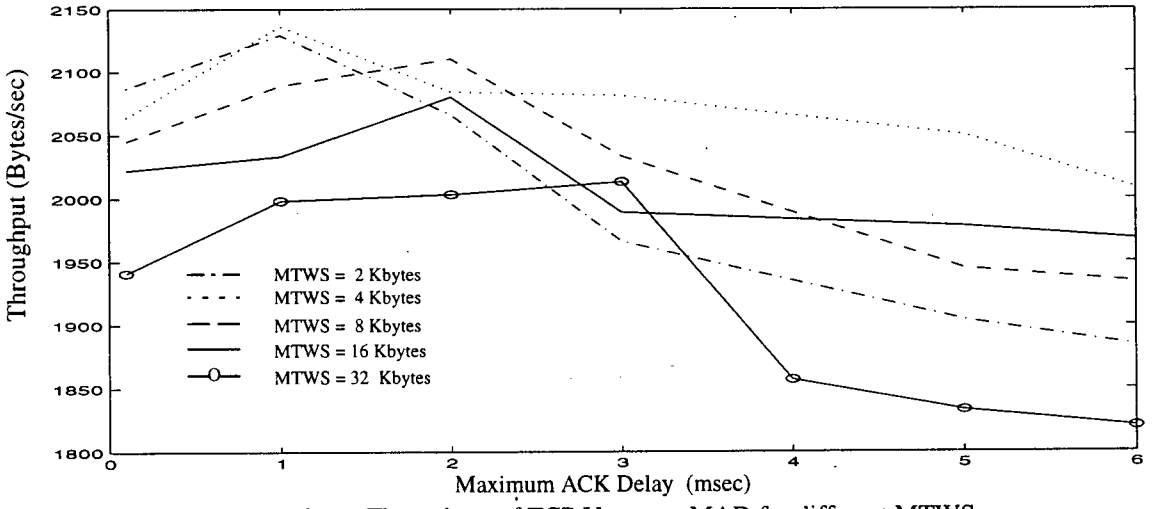

Figure 5.19 Throughput of TCP Vegas vs. MAD for different MTWSs

Figure 5.18 and figure 5.19 show that a smaller window size gives better throughput for both TCP Reno and TCP Vegas over a whole range of the MADs. When the MAD is increased, their throughput is increased. Beyond a certain ACK delay, their throughput is gradually decreased because the congestion window in the sender grows too slowly to bring the throughput back to the norminal level before the next handoff occurs. A smaller MTWS and a shorter MAD, give a much higher throughput for both TCP implementations. However, with a larger MTWS, particularly for TCP Reno, a longer MAD is needed for good throughput performance because the longer MAD reduces the data transmission rate at the sender. The optimal MADs and the corresponding TCP throughput under different conditions are summarized in Table 5.2. From table 5.2, TCP Vegas, overall, has better throughput performance than TCP Reno.

Table 5.2 The optimal MADs for TCP Reno and TCP Vegas

	TCP Vega	TCP Reno
ACK delay at mobile speed = 45 Km/h and transmit window size = 16 Kbytes	1 msec	2 msec
Throughput (Bytes/sec)	2165	2039
ACK delay at mobile speed = 75 Km/h, 110 Km/h and transmit window size = 16 Kbytes	3 msec	3 msec
Throughput (Bytes/sec)	2101, 2094	2020, 2009
ACK delay, transmit window size = 4 bytes	1 msec	2 msec
Throughput (Bytes/sec)	2136	2019
ACK delay, transmit window size = 16 bytes	2 msec	3 msec
Throughput (Btytes/sec)	2080	2019

5.6.3 Effects of Maximum Segment Size

Since the PLR in the bad state within the normal state is directly depended on the BER and segment size, the net effect of the segment size for the specific BER is to change the PLR in the bad state, which effects on TCP have been considered in figure 5.5 in section 5.4. If the segment

size increases at a specified BER, the PLR is increased almost linearly. However, in this wireless environment, propagation conditions may render the wireless channel unavailable for a short period of time or mobile users needs to be handoffed when they move out of one cell to another cell. Therefore, the PLR over a mobile channel not only depends on the PLR in the bad state within the normal state but also depends on the mobility rate and the handoff delay.

In order to maximize TCP throughput over the mobile radio channel, this subsection investigates the effects of the MSS on the throughput of TCP Reno and TCP Vegas in the MRCII environment at different mobile speeds for different PLRs in the bad state. All other simulation parameters are fixed, but the mobile speed is varied for different MSSs in the first scenario and the 1/a is varied for different MSSs in the second scenario. The MTWS is assumed to be 8 Kbytes. The MAD is assumed to be 1 msec. The BER in the bad state within the normal state is fixed at $1x10^{-6}$ (a typical BER in wireless channels). In the first scenario, the radius of the radio cells and the area of the handoff region are fixed at 2 Km and 100 m, respectively. In the second, the mobile speeds is fixed at 75 Km/h. Figure 5.20 shows the throughput of TCP Reno an TCP Vegas as a function of the MSS at different mobile speeds. Figures 5.21 and 5.22 show the throughput of TCP Reno and TCP Vegas, respectively, as a function of 1/a for different MSSs.

Figure 5.20 shows that the MSS has a greater effect on the throughput of TCP Vegas than on that of TCP Reno at different mobile speeds. At the mobile speeds of 45 Km/h and 75 Km/h, the throughput of TCP Reno is gradually increased when MSS increased to a certain value and then decreases. At a mobile speed of 110 Km/h, the throughput of TCP Reno is hardly changed when the MSS is increased. A relatively small MSS such as 256 bytes, gives the worst throughput for TCP Vegas at different mobile speeds. At a lower mobile speed, the throughput of TCP Vegas is first sharply increased and then gradually decreased when the MSS is increased. At a mobile speed of 110 Km/h, a larger MSS seems to give a higher throughput in TCP Vegas.

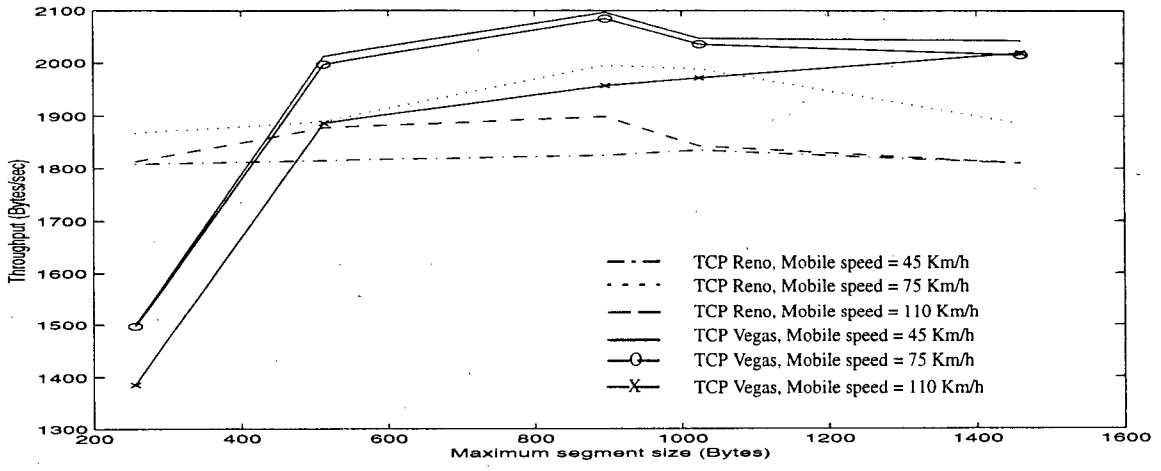

Figure 5.20 TCP throughput vs the MSS for different mobile speeds

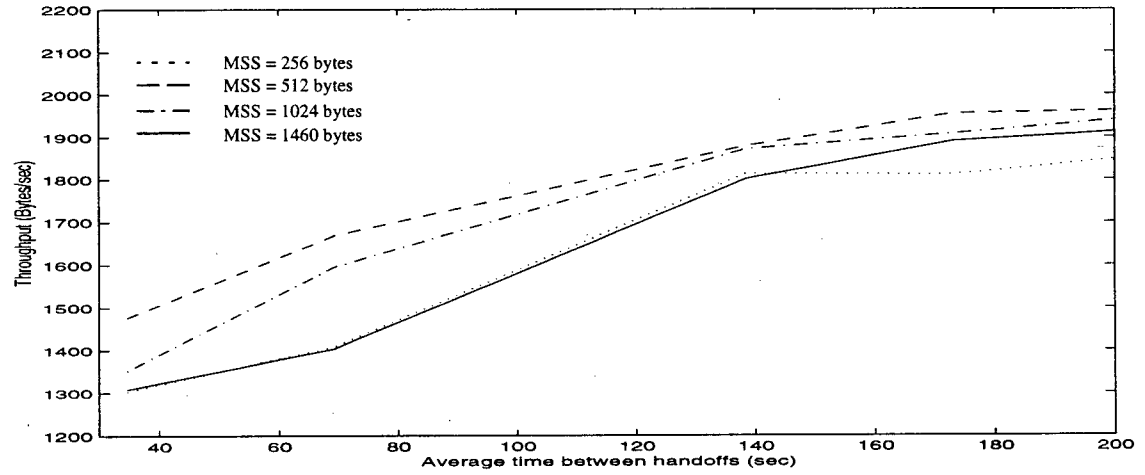

Figure 5.21 The throughput of TCP Reno vs 1/a for different MSSs

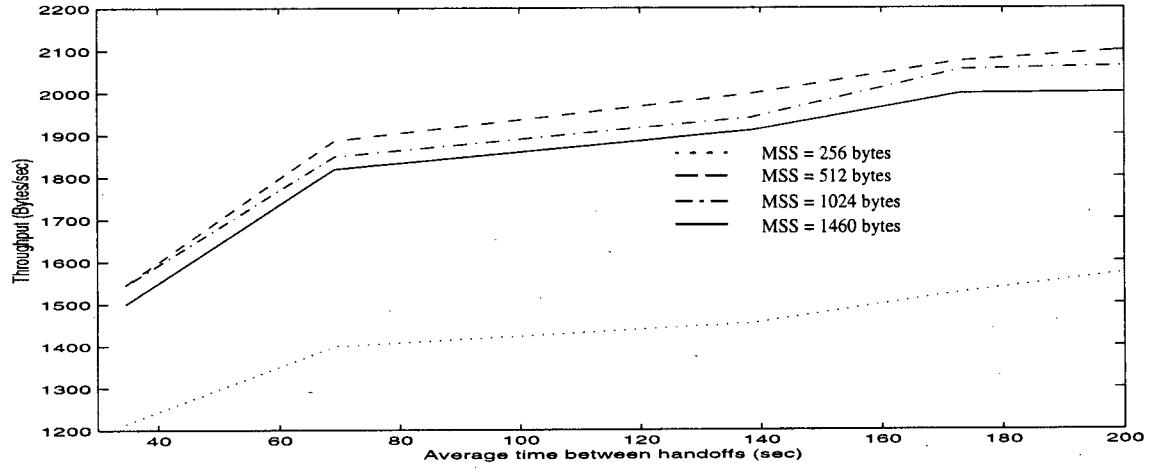

Figure 5.22 The throughput of TCP Reno vs 1/a for different MSSs

Figures 5.21 and 5.22 show that the throughput of TCP Reno and TCP Vegas is increased when 1/a is increased for different MSSs. However, too small a MSS may significantly reduce the amount of data sent per each full sending window, thus degrading TCP throughput. On the other hand, too large a MSS may increase the probability of error per packet, thus also degrading TCP throughput. At the mobile speed = 75 Km/h and 1/b = 3.5 sec, both TCP implementations with a 512 bytes MSS have better throughput than that with the other MSSs; whereas with a 256 byte MSS, particular for TCP Vegas, gives the worst throughput over the whole range of the 1/a values. Table 5.3 compares the throughput of both TCP Reno and TCP Vegas as a function of the MSS for different BERs in the bad state, at 1/b = 2 sec, 1/a = 60 sec, and the mobile speed = 75 Km/h.

Table 5.3 The Throughput of TCP Reno and TCP Vegas (Bytes/sec)

Maximum segment size	TCP Reno BER in the bad state = $1x10^{-6}$	TCP Vegas BER in the bad state = $1x10^{-6}$	TCP Reno BER in the bad state = $5x10^{-5}$	TCP Vegas BER in the bad state = $5x10^{-5}$
256	1771.4	1361.8	1570.4	843.1
384	1689.8	1760.2	1345.3	1333.7
512	1887.5	2003.7	1270.7	1315.2
640	1851.1	2083.2	1119.5	1178.8
768	1840.6	2094.8	1178.2	1249.5
896	1912.9	2168.1	955.16	1090.9
1024	2021.1	2188.5	901.9	925.3
1460	2020.1	2189.4	763.3	795.4

Table 5.3 shows that a larger MSS has a higher throughput performance in both TCP Reno and TCP Vegas for a relatively low PLR in the bad state. It is because a larger MSS can carry more data for every round-trip time before the next channel interruption occurs. However, at a higher PLR, a smaller MSS seems to give higher throughput because the PLR affects not only data packets but also ACK packets. Because of the detail calculation of RTT and RTO for every TCP segment in TCP Vegas, TCP Reno has much better throughput than TCP Vegas when a MSS

of 256 Kbytes or less is used. Otherwise, TCP Vegas has better throughput than TCP Reno for the whole range of the MSSs at different mobile speeds for different 1/a values.

5.7 Optimization and Discussion

In order to improve throughput performance of TCP Reno and TCP Vegas in the MRCII environment under most conditions, this section provides a detailed evaluation for jointly optimizing several TCP parameters including the MTWS, the MAD, and the MSS. From the simulation results in section 5.6, an upper bound and a lower bound of the TCP parameters are obtained and suggested for both TCP implementations under different conditions. In this section, all other simulation parameters are fixed, but the above TCP parameters are varied under different conditions. The RTT coefficient deviation is assumed to be 4. The thresholds (α and β) in TCP Vegas are assumed to be 1 and 3. The BER in the bad state within the normal state is fixed at $1x10^{-6}$. The area of the handoff region is fixed at 100 m. When the mobile host moves from one cell to another cell, the wireless channel is assumed to be unavailable for 1/b sec every 1/a sec.

Figures 5.23-5.26 show the throughput of TCP Reno with different MTWSs and MADs as a function of 1/a for different MSSs, respectively, at a mobile speed of 75 Km/h. From those figures, for a MSS > 256 bytes, TCP Reno with a 4 Kilobyte MTWS and a smaller MAD seems to have much higher throughput than the other MTWSs over the whole range of the 1/a values. For a MSS of 256 bytes, a 2 Kilobyte MTWS seems to give higher throughput for TCP Reno when 1/a < 180 sec. TCP Reno with a 4 Kilobyte MTWS and a longer MAD seems to have much higher throughput than the other MTWSs at a longer 1/a. However, a 2 Kilobyte MTWS seems to give higher throughput than the other MTWSs at a shorter 1/a. For different MADs and different MSSs, a 32 Kilobyte MTWS seems to give much less throughput than the other MTWSs over the whole range of the 1/a values.

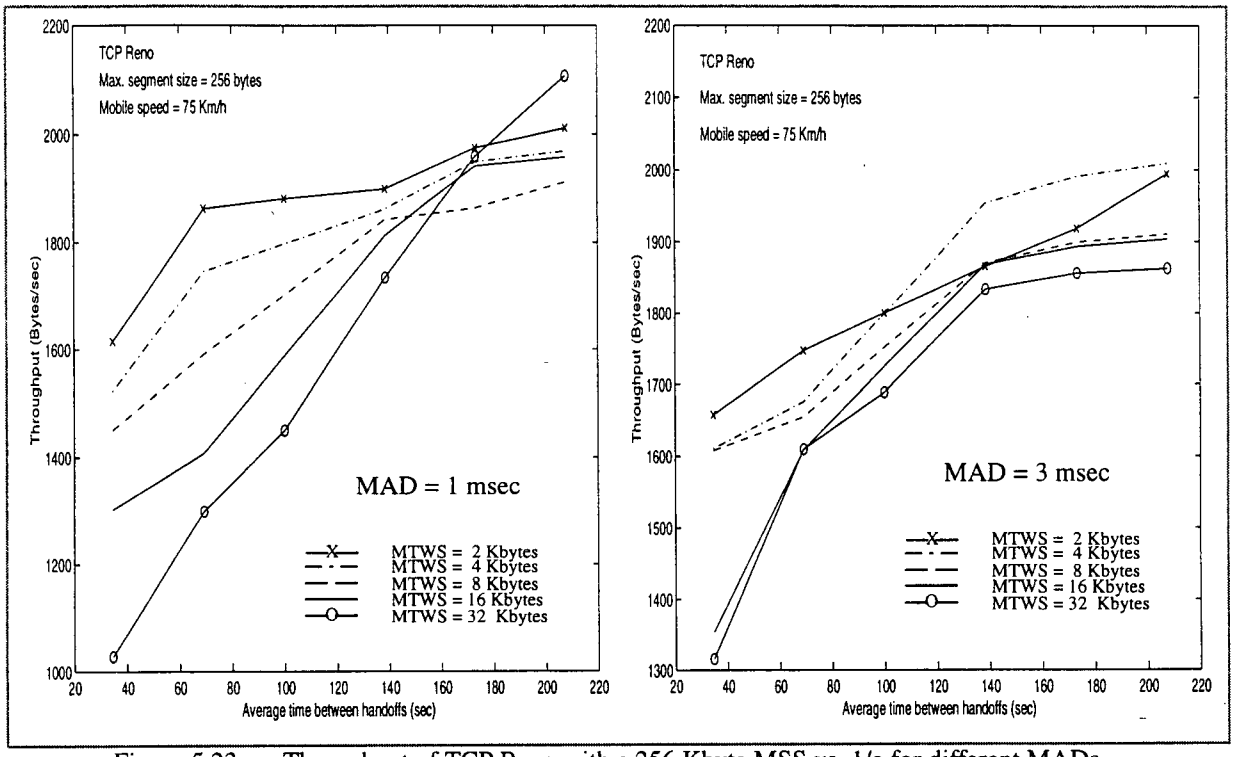

Figure 5.23 Throughput of TCP Reno with a 256 Kbyte MSS vs. 1/a for different MADs

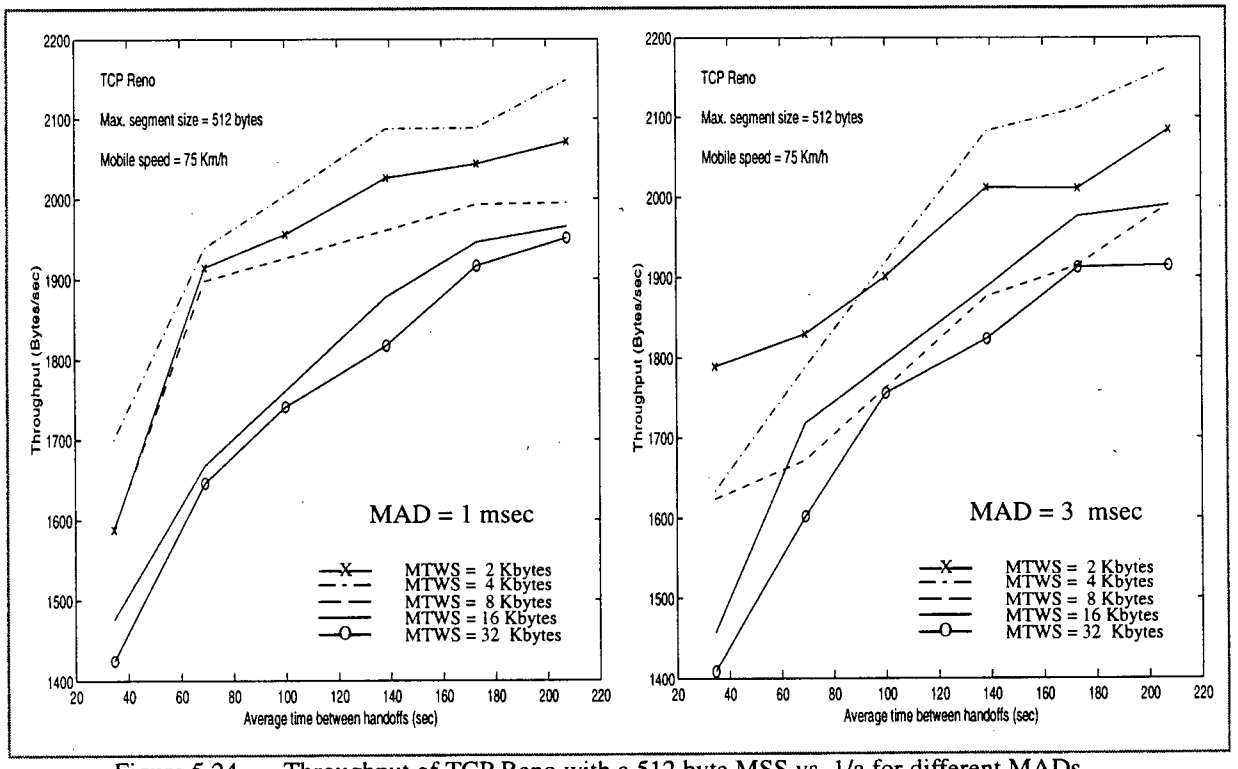

Figure 5.24 Throughput of TCP Reno with a 512 byte MSS vs. 1/a for different MADs

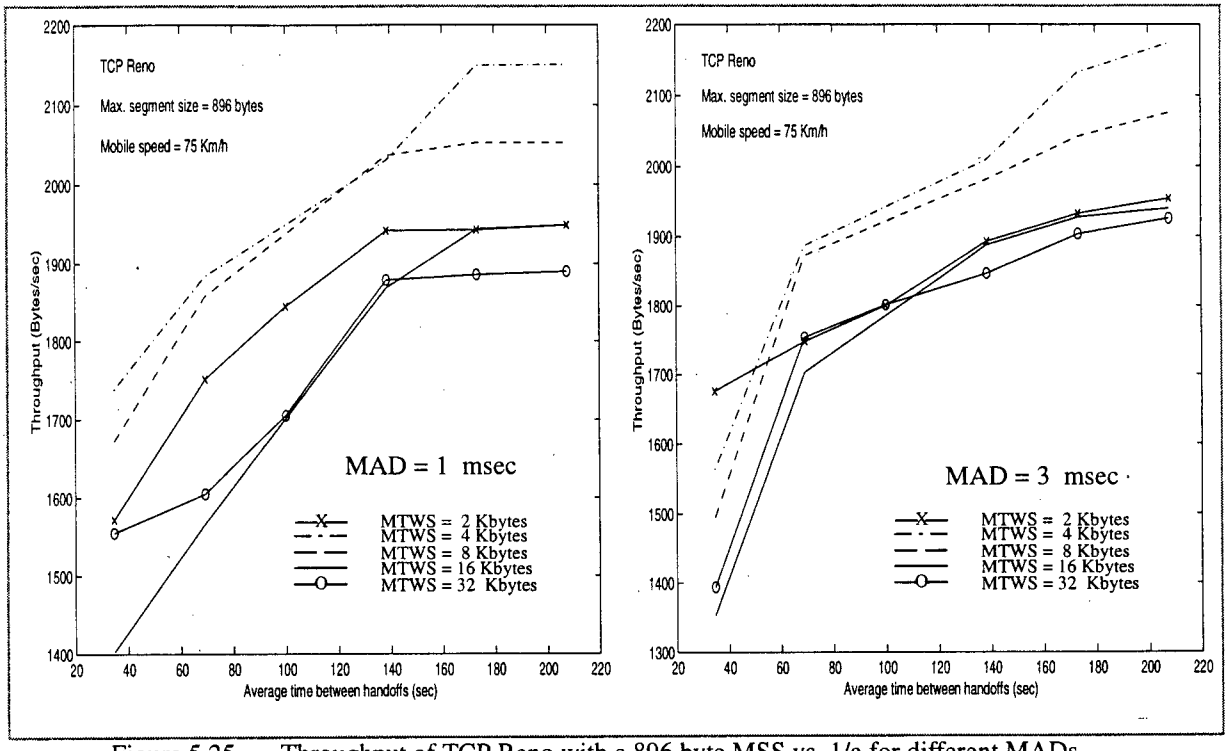

Figure 5.25 Throughput of TCP Reno with a 896 byte MSS vs. 1/a for different MADs

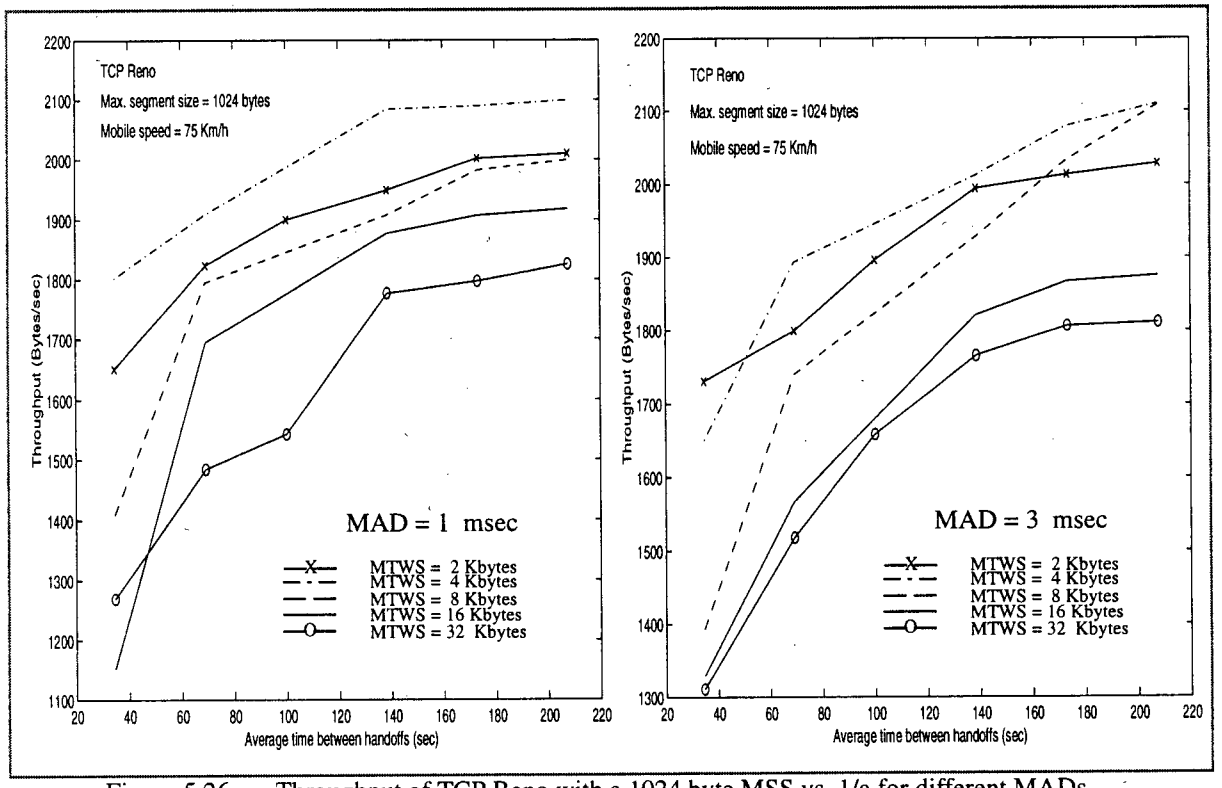

Figure 5.26 Throughput of TCP Reno with a 1024 byte MSS vs. 1/a for different MADs

From figures 5.23-5.26, for different MADs and different MTWSs, a MSS within a range between 512 bytes to 1024 bytes (e.g. 896 bytes in this case) seems to give higher throughput for TCP Reno than other MSSs over the whole range of the 1/a values; whereas a 256 byte MSS seems to give the worst throughput for TCP Reno than other MSSs. For different MSSs and different MTWSs, a 1 millisecond MAD seems to give higher throughput for TCP Reno than other MADs particular in a high mobility environment. Over a whole range of 1/a, a 4 Kilobyte MTWS gives reasonably high throughput for TCP Reno for different MSSs and different MADs.

Figure 5.27-5.30 show at a mobile speed of 75 Km/h, the throughput of TCP Vegas with different MTWSs and different MADs as a function of 1/a for different MSSs, respectively. From those figures, For a shorter MAD and a MSS $\leq$ 512 bytes, a 2 Kilobyte MTWS gives much higher throughput for TCP Vegas than the other MTWSs over the whole range of the 1/a. However, a 4 Kilobyte MTWS seems to give higher throughput for TCP Vegas for a shorter MAD and a MSS > 512 bytes. For a longer MAD and a longer 1/a, a 4 Kilobyte MTWS gives much higher throughput for TCP Vegas than the other MTWSs for different MSSs. However, a 2 Kilobyte MTWS and a smaller MSS give slightly better throughput for TCP Vegas than the other MTWSs for a shorter 1/a. For different MADs and different MSSs, a 32 Kbyte MTWS gives the worst throughput for TCP Vegas than the other MTWSs over the whole range of the 1/a.

Overall, for different MADs and different MTWSs, a 896 byte MSS seems to give reasonably high throughput performance for TCP Vegas over the whole range of the 1/a; a 256 byte MSS seems to give the worst throughput performance for TCP Vegas. For different MSSs and different MTWSs, a 1 millisecond MAD seems to give higher throughput performance for TCP Vegas than other MADs over the whole range of the 1/a, particular in the high mobility environment. For different MSSs and different MADs, a 8 Kbyte MTWS give reasonably high throughput performance for TCP Reno over the whole range of the 1/a.

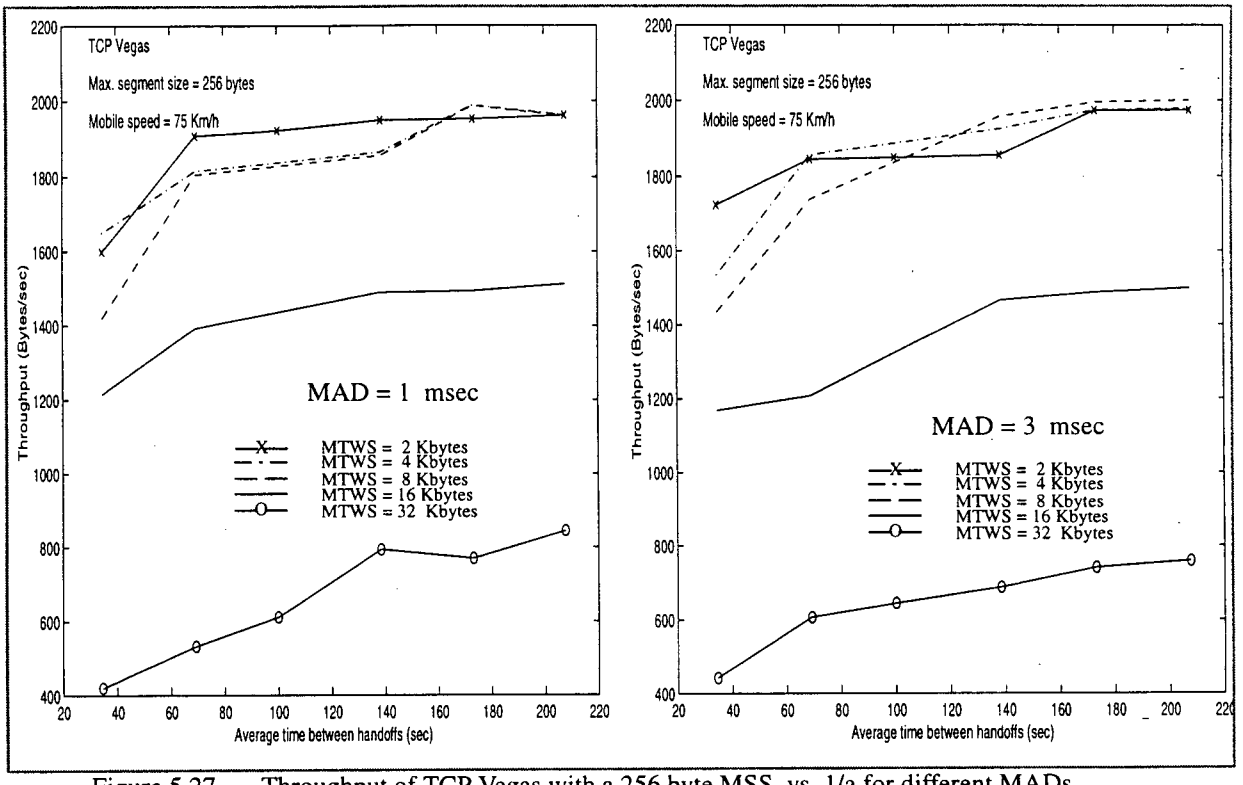

Figure 5.27 Throughput of TCP Vegas with a 256 byte MSS vs. 1/a for different MADs

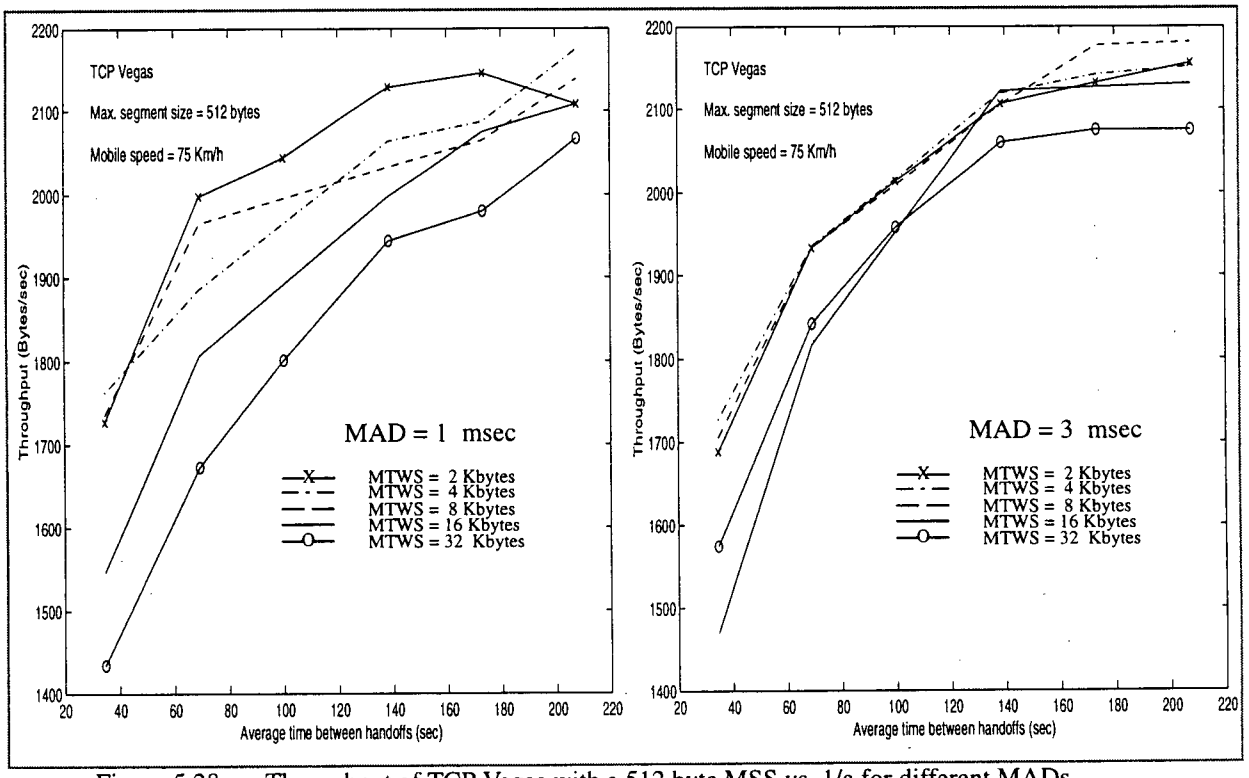

Figure 5.28 Throughput of TCP Vegas with a 512 byte MSS vs. 1/a for different MADs

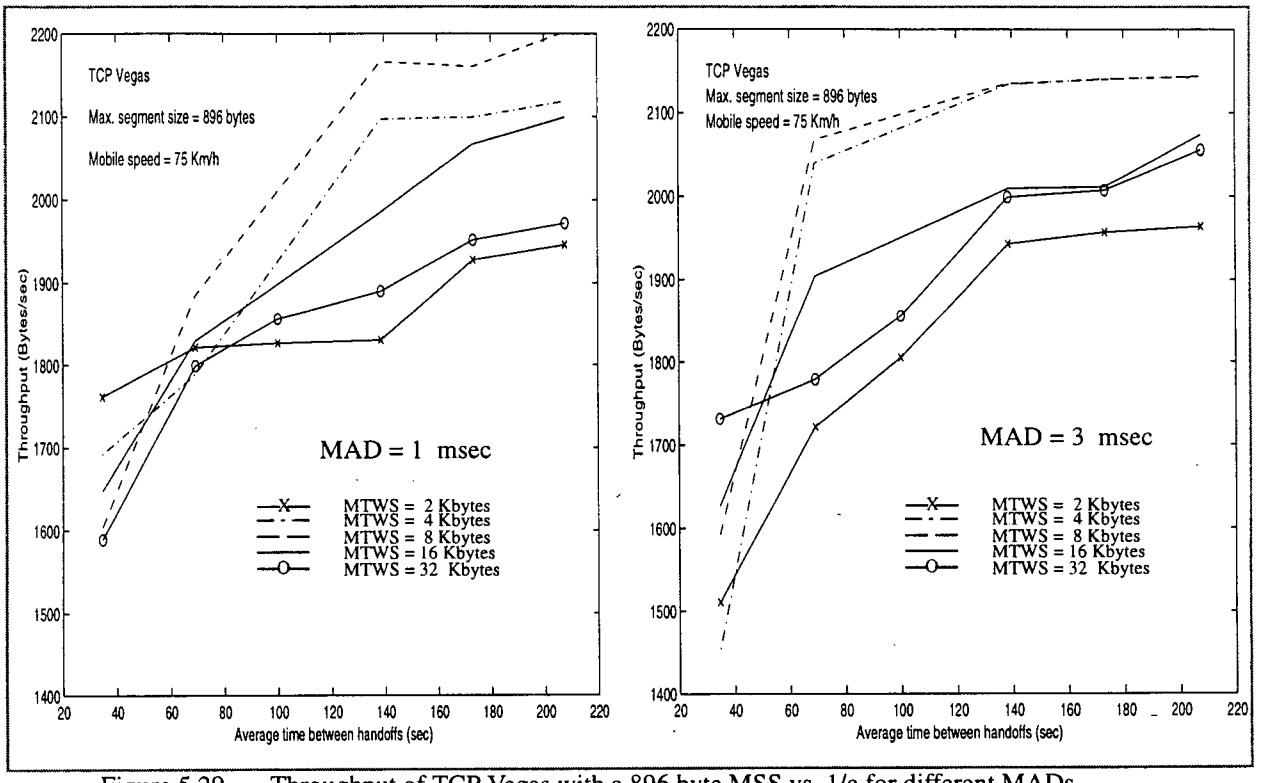

Figure 5.29 Throughput of TCP Vegas with a 896 byte MSS vs. 1/a for different MADs

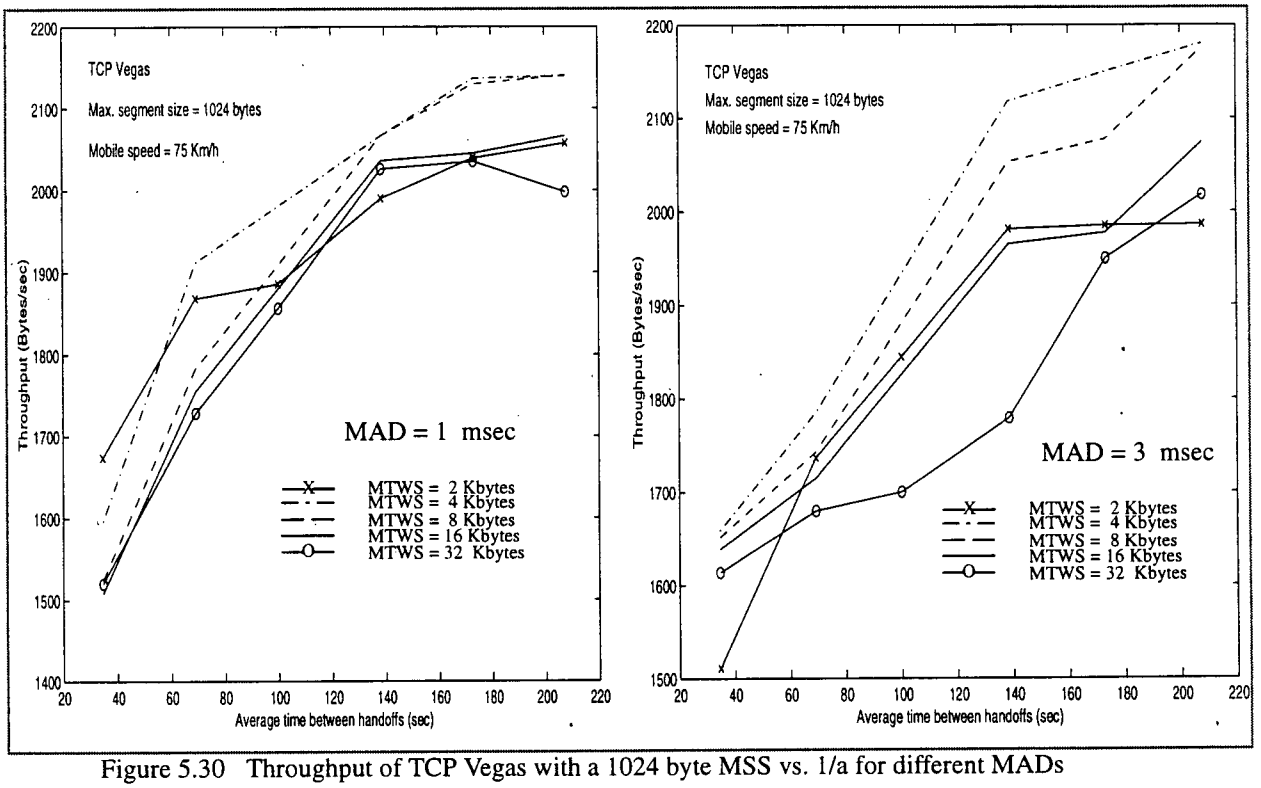

Figure 5.30 Throughput of TCP Vegas with a 1024 byte MSS vs. 1/a for different MADs

Based on a series of simulations to jointly optimize the TCP parameters for both TCP Reno and TCP Vegas at different mobile speeds and for different 1/a values. Table 5.4 summarizes the suggested optimal values of the chosen TCP parameters for TCP Vegas and TCP Reno under most conditions for the MRCII environments. Figure 5.31 shows the throughput of both TCP implementations with the optimal TCP parameters as a function of 1/a at a mobile speed = 75 Km/h, 1/b = 3.5 sec and the PLR in the bad state within the normal state = 0.04.

Table 5.4 The jointly optimal TCP parameters for TCP Vegas and TCP Reno

	Max. Window Size	**Max. ACK Delay (msec)**	**Segment Size x (Bytes)**
TCP Vegas	8 Kbytes	1	896 < x <1024
TCP Reno	4 Kbytes	1	896 < x <1024

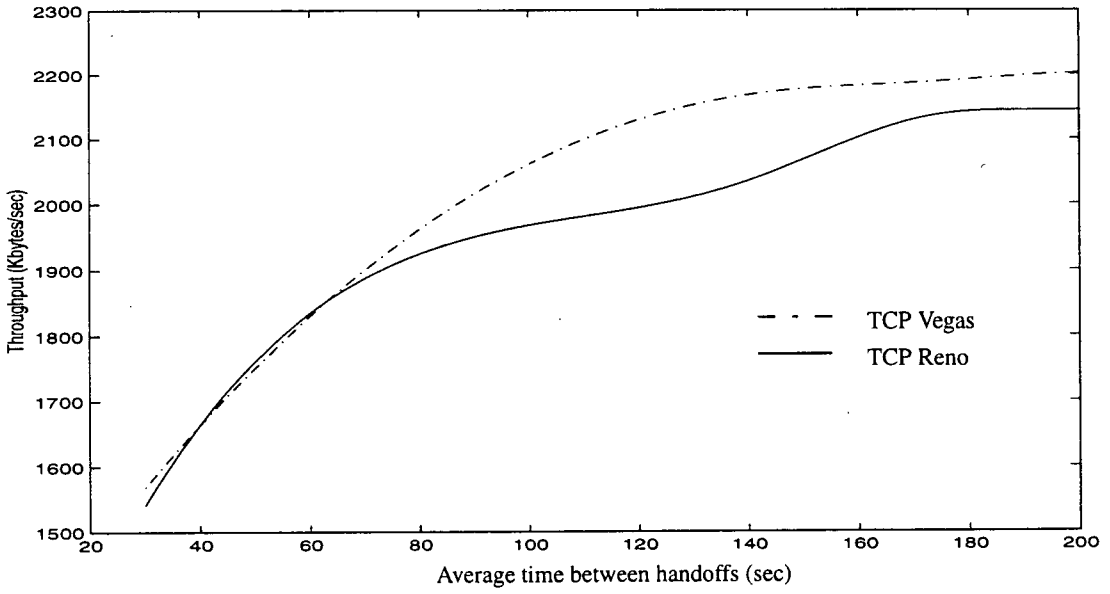

Figure 5.31 TCP throughput vs. 1/a

Compared with figure 5.10, figure 5.31 shows that the throughput of TCP Reno is significantly improved after jointly optimizing the TCP parameters particular in the low mobility environment. However, the throughput of TCP Vegas is slightly improved after optimizing the

TCP parameters over the whole range of the 1/a. After jointly optimizing the TCP parameters, TCP Vegas still has better throughput than TCP Reno under most conditions. With the optimal TCP parameters, TCP Vegas has approximately the same throughput as TCP Reno in a relatively high mobility environment; however, TCP Vegas has a much higher throughput than TCP Reno in a relatively low mobility environment. Overall, TCP Vegas has approximately 0.1% to 8% better throughput than TCP Reno when 1/a varies from 35 sec to 200 sec.

Chapter 6 Summary and Conclusions

The interest in the performance analysis of TCP implementations over wireless channels interconnected with the Internet is motivated by the critical need for the desired quality of service in seamless ETE delivery of the data traffic over a HWW network. Previous studies have demonstrated the poor performance of both TCP RFC and TCP Reno in networks with satellite links or with mobile links. This thesis focuses on the performance evaluation of TCP RFC, TCP Reno, and TCP Vegas in networks with satellite links and mobile links interconnected with remote LANs over the Internet. It also studies the effects of the TCP parameters on the TCP throughput performance, while attempting to improve the TCP throughput performance by jointly optimizing the TCP parameters for both HWW networks.

6.1 Summary of Findings

The throughput performance of the different TCP implementations in networks with satellite links and mobile links interconnected with the Internet has been thoroughly analyzed and compared. In the BSII environment, the effects of the network elements including the PLR in the satellite links, the STPs between the clear sky state and the rain fade state and the average ETE delay on TCP throughput has been studied. In the MRCII environment, the effects of the network elements including the STPs among the three states (the handoff state, the good state and the bad state within the normal state), the mobile fading, mobility rates, and handoff interruptions on TCP throughput has also been studied. The effects of the chosen TCP parameters including the MTWS, the MAD, and the MSS on TCP throughput in both HWW networks have been investigated, while the optimal TCP parameters for the different TCP implementations in both HWW networks have also been obtained.

From the simulation results, the chosen TCP parameters strongly affect the throughput of the different TCP implementations in networks with satellite links and mobile links interconnected with the Internet with limited resources. Too big MTWSs and too small MADs increase network congestion and cause network buffer overflow; whereas too small MTWSs and too large MADs result in inefficient bandwidth utilization. Too large MSSs increase the probability of packet error; whereas too small MSSs cause inefficient bandwidth utilization.

In the presence of the long propagation delays and high PLRs of satellite links, the three different TCP implementations cannot have efficient bandwidth usage as TCP throughput oscillates with the congestion window. The PLR in the satellite link, the ETE delay, and the chosen TCP parameters have much greater effect on the throughput of TCP Vegas than on that of TCP Reno and TCP RFC; whereas those network elements only have a slight effect on the throughput of TCP RFC. The big window option works well and significantly improves the throughput for both TCP Reno and TCP Vegas only at a relatively low PLR but seriously degrades the throughput of TCP RFC. Because of the limitations of the Internet and the trade-off of a bigger MTWS (creating its own losses by overloading the network), both TCP Reno and TCP RFC prefer to use a smaller MTWS and a shorter MAD, particular in high PLR networks, in order to reduce the probability of congesting the network and to have a faster error recovery. However, because of its better congestion avoidance mechanism and use of a finer-grained timer to update the current RTO for every segment, TCP Vegas prefers to use a bigger MTWS and a longer MAD in order to send more data packet per full window size and have more accurate information in the returned acknowledgment. A larger MSS can carry more data for every round-trip time and can fill up a long satellite pipe more efficiently, thus improving the TCP throughput in network with satellite links.

The optimal TCP parameters for the different TCP implementations are summarized in Table 4.4. After jointly optimizing the chosen TCP parameters, TCP RFC is still much inferior in throughput to TCP Reno and TCP Vegas under most conditions; whereas TCP Vegas has much higher throughput than the others. TCP Vegas, overall, has an approximately 23% to 100% better throughput than TCP Reno and 500% to 700% better throughput than TCP RFC as the PLR in the satellite links varies from 0.001 to 0.06. TCP Vegas has nearly 40% better throughput than TCP Reno and more than 700% better throughput than TCP RFC over the whole range of the ETE delays.

In the presence of high BERs, high mobile speeds, intermittent connectivity, and handoff interruptions in the mobile radio channel, both TCP Reno and TCP Vegas cannot perform well in the MRCII environment because these network elements completely violate most of TCP's operational assumptions. The network elements and the chosen TCP parameters have great effects on their throughput performances. At relatively high mobile speed, they prefer to use a smaller MTWS and a shorter MAD. At a relatively low mobile speeds, they prefer to use a bigger MTWS and a longer MAD. For different 1/a values and different 1/b values, a smaller MTWS and a shorter MAD give a higher throughput for both TCP implementations. A larger MSS seems to have relatively high throughput efficiency for both TCP implementations even under high mobility conditions.

The jointly optimal TCP parameters for both TCP Reno and TCP Vegas are summarized in Table 5.4. After jointly optiminzing TCP parameters, the throughput of TCP Vegas is overall better than TCP Reno under most conditions. TCP Vegas has an approximately 0.1% to 8% better throughput than TCP Reno when the average time between handoffs varies from 35 sec to 200 sec at the PLR in the bad state = 0.04, the mobile speed = 75 Km/h, and 1/b = 3.5 sec.

6.2 Future Investigation

Based on the following main reasons, network congestion is unlikely and it is reasonable for a proposed TCP implementation to assume by default that any loss is due to medium errors. First, most of the losses in the Internet occur one packet at a time [32]. Second, the simulation results show that most of the losses are due to media error if a better congestion avoidance mechanism is used. The coarse-grain RTO and the exponential backoff algorithm seriously degrade the TCP throughput performance. Third, in today's high speed networks, network bandwidth is carefully managed on private links.

The proposed TCP (P-TCP) uses the TCP Vegas variant of the Slow Start algorithm and the congestion avoidance algorithm in TCP Vegas. P-TCP uses the fine-grain RTO algorithm in TCP Vegas to retransmit lost segments, but it does not apply the coarse-grain RTO and the exponential backoff algorithm. Simulation results show that P-TCP has an approximately 6% to 20% better throughput than TCP Vegas as PLR in the satellite links varies from 0.005 to 0.05 (see figure 6.1). It also has an approximately 3% to 30% better throughput than TCP Vegas as the PLR in the bad state varies from 0.005 to 0.1 (see figure figure 6.2). However, the multiple packet losses per each full sending window size, still seriously degrade TCP throughput. Therefore, subject to the continued interest in the performance improvement of TCP for networks with satellite links and mobile links interconnected with the Internet, further research and more analysis may be needed. The following areas are suggested for further investigation:

- A better TCP acknowledgment method in the receiver to inform the sender to send all the lost packets without overflowing the buffers

- A better retransmission method to enable the sender to discriminate packet losses due to the media from packet losses due to congestion.

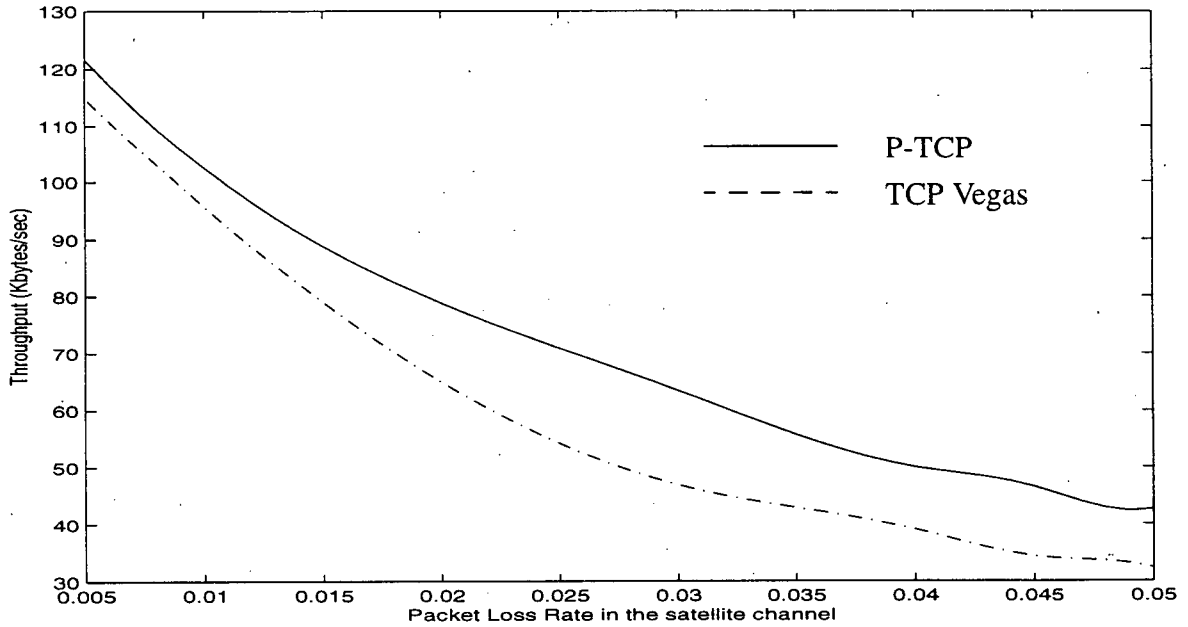

Figure 6.1 The throughput comparison of TCP Vegas and a proposed TCP vs. the PLR in the satellite links

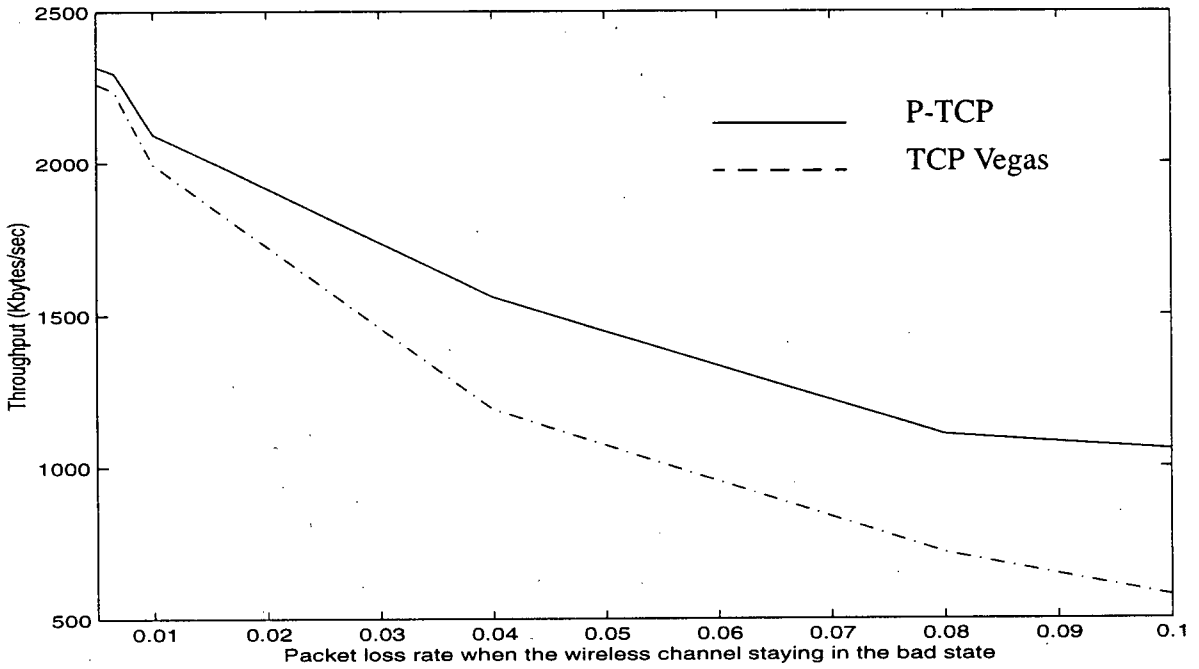

Figure 6.2 The throughput comparison of TCP Vegas and a proposed TCP vs. the PLR in the bad state within the normal state for networks with mobile links

Bibliography

[1] M. Schwartz, "Network Management and Control Issues in Multimedia Wireless Networks", *IEEE Personal Communications*, pp. 8-16, June 1995.

[2] M. Talla, A. K. Elhakeem and M. Kadoch, "Throughput and Delay Characteristics of ARQ Transport Protocols over LEO and GEO Satellite Networks", *Singapore ICCS 94*, pp. 410-414, 1994.

[3] M. Aghadavoodi Jolfaei, B. Heinrichs, and M. R. Nazeman, "Improved TCP Error Control for Heterogeneous WANs", *IEEE National Telesystems Conference 94*, pp. 257-260, July 1994.

[4] Oliver W. W. Yang, Xiao-Xiong Yao, and K. M. S. Murthy, "Modeling and Performance Analysis of File Transfer in a Satellite Wide Area Network", *IEEE Journal on Selected Areas in Communications*, Vol. 10, No. 2, pp. 428 -436, Feb. 1992.

[5] H. Linder, W. Stering, U. Hofmann, "Performance of XTPX and TCP/IP in a satellite environment", *Proceedings of 20th Conference on Local Computer Networks*, pp. 246-253, October 1995.

[6] P. Manzoni, D. Ghosal, G. Seazzi, "A simulation Study of the Impact of Mobility on TCP/IP", *Network Protocol, 1994 International Conference*, pp. 196-202, May 1994.

[7] R. Caceres and L. Iftode, "Improving the Performance of Reliable Transport Protocols in Mobile Computing Environments", *IEEE Journal on Selected Areas in Communications*, pp. 850-857, June 1995.

[8] M. C. Chuah, O. C. Yue and A. DeSimone, "Performance of Two TCP Implementations in Mobile Computing Environments", *IEEE Journal on Selected Areas in Commumications*, pp. 339-344. June 1995

[9] Z. Haas and Chih-Lin I, "On Handoffs in Packetized Wireless Systems", *Globecom '93: IEEE Global Telecommunications Conference*, pp. 1808-1814, June 1993.

[10] H. Balakrishnan, S. Seshan, and R. H. Katz, "Improving Reliable Transport and Handoff Performance in Cellular Wireless Network", *ACM Mobile Computing and Networking Conference, Berkeley, California*, pp. 14-15, November 1995.

[11] P. Karn and C. Partidge, "Improving Round Trip Time Estimates in Reliable Transport Protocols", *ACM Transactions on Computer Systems*, pp. 364-373, Sept. 1991.

[12] Protocol Engines, Inc., "XTP Protocol Definition", *Revision 3.6, XTP Forum*, Jan. 1992.

[13] A. Bakreand B. R. Badrinath, "I-TCP: Indirect TCP for Mobile Host", *Technical Report DCS-TR-314, Rutgers University,* Oct. 1994.

[14] E. Amir, H. Balakrishnan, S. Seshan, and R. H. Katz, "Efficient TCP over Networks with Wireless Links", *Proc. ACM Mobicom*, Nov. 1995

[15] A. DeSimone, M. C. Chuah, and O.C. Yue, "Throughput Performance of Transport-Layer Protocols over Wireless LANs", *Proc. Globecom '93*, Dec. 1993.

[16] H. K. Randy, "Adaption and Mobility in Wireless Information Systems", *Proc. ACM Mobicom*, Nov. 1995.

[17] V. Jacobson, "Congestion Avoidance and Control", *Proceeding of ACM SIGCOMM' 88*, Aug. 1988.

[18] W. Stevens, "TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms", Request for Comments 2001, IETF, January 1997.

[19] V. Jacobson and R.T. Braden, "TCP Extensions for Long Delay Paths", Request for Comments 1072, IETF. October, 1988.

[20] V. Jacobson, R. T. Braden, and D. A. Borman, "TCP Extensions for High Performance", Request for Comments 1323, IETF, 1988.

[21] R. Fox, "TCP Big Window and Nak Options", Request for Comments 1106, IETF, June, 1989.

[22] V. Paxson, "Empirically Derived Analytic Models of Wide-Area TCP Coonnection", *IEEE Transactions on Networking*, Vol. 2, No. 4, August 1994.

[23] R. Caceres, P. B. Danzig, S. Jamin, D. J. Mitzel, "Characteristics of Wide-Area TCP/IP Conversations", *ACM SIGCOMM' 91,* pp. 103-110, May 1991.

[24] V. Paxson and S. Floyd, "Wide Area Traffic: The Failure of Poisson Modeling", *IEEE/ ACM Transactions on Networking*, Vol. 3, No. 3. June 1995

[25] K. M. Khalil, Y. S. Sun, "Peformance Considerations for TCP/IP in Wide Area Networks", *Local Computer Networks, 1994 19th Conference*, pp. 166-175, March 1993

[26] L. S. Brakmo and L. L. Peterson, "TCP Vegas: end to end congestion avodiance on a global Intenet", *IEEE Journal on Selected Areas in Communications*, Vol 13, No. 8. pp. 1465-1480, October 1995.

[27] W. R. Stevens, *TCP/IP Illustrated, Volume 1: The Protocols.* Addison-Wesley, 3rd ed., 1994.

[28] J. B. Postel, "Transmission Control Protocol", Request for Comments 793, IETF, Sept., 1981.

[29] R. F. Quick, "An overview of the cellular digital packet data system", *Proceedings of the Fourth International symposium on Personal, Indoors and Mobile Radio Communications,* pp. 338-343, Sept. 1993.

[30] R. Jain, *The art of computer systems performance analysis: techniques for experimental design, measurement, simulation, and modeling*, Wiley, pp. 491, 1991.

[31] M. Filip, E. Vilar, "Optimum utilization of the channel capacity of a satellite link in the presence of amplitude scintillations and rain attenuation", *IEEE Transactions on communications*, vol.38, No. 11, pp. 1958-1965, November 1990.

[32] A. Sanghi, A. K. Agrawala, and O. Gudmundsson, " Experimental Assessment of End-to-End Behavior on Internet", *Computer Communication Proceeding, IEEE INFOCOM'93,* pp. 98-115, Nov. 1993.

[33] *Cellullar Digital Packet Data System Specification.* CDPD Forum, Inc. Release1.1. Jan. 1995.

[34] S. Jangi and L.F. Merakos, "Performance Analysis of Reservation Random Access Protocols for Wireless Access Network," I*EEE Transaction on Comm.* Vol 42, pp. 1223-1234, April, 1994.

[35] D. Anick, D. Mitra, and M. M. Sondhi, "Stochastic theory of a data-handling system with multiple sources," *Bell System Technical Journal*, Vol. 61, pp. 1871-1894, 1982

[36] W. C. Y. Lee, *Mobile Cellular Telecommunications Systems*, McGraw-Hill Inc., 2nd ed., 1995.

[37] K. Khalil, J. Hand, and M. Mariswamy, "Analysis and Traffic Characterization of a Wide Area Network," *Proceedings of ICC'93, Geneva*, pp. 23-26, May 1993.

[38] G. Arnold and T. W. Kao, "Rain Attenuation in EHF Satellite Communications", *IEEE Pacific Rim Conference on Communications, Computers, and Signal Processing. Proceedings,* pp. 153-156, May, 1995.

[39] Wolfhard J. Vogel, Geoffrey W. Torrence, and Jeremy E. Allnutt, "Rain Fades on Low Elevation Angle Earth-Satellite Paths: Comparative Assessment of the Austin, Texas, 11.2-GHz Experiment", *Poceedings of the IEEE*, Vol. 81, No. 6, pp. 885-896, June 1993

Appendix A. List of Abbreviations and Acronyms

ACK	Acknowledgment
ARQ	Automatic Repeat Request
BER	Bit Error Rate
BS	Base Station
BSII	Broadband Satellite Interworking with the Internet
CDMA	Code Division Multiple Access
CDPD	Cellular Digital Packet Data
ETE	End-to-End
FRFR	Fast retransmit and fast recovery
FTP	File Transfer Protocol
GEO	Geosychronous
HWW	Heterogeneous Wireless and Wired
IP	Internet Protocol
ICMP	Internet Control Message Protocol
IGMP	Internet Group Management Protocol
LAN	Local Area Network
LEO	Low Earth Orbit
NNTP	Network News Transfer Protocol
MAD	Maximum ACK delay
MRCII	Mobile Radio Channel Interworking with the Internet
MSS	Maximum Segment Size
MTWS	Maximum Transmit Window Size

MTU	Maximum Transfer Unit
MDIS	Mobile Data Intermediate System
NAK	Negative Acknowledgment
PLR	Packet Loss Rate
RTT	Round Trip Time
RTO	Retransmission Time Out
TCP	Transmission Control Protocol
TDMA	Time Division Multiple Access
UDP	User Datagram Protocol

Appendix B. Opnet Simulation Models

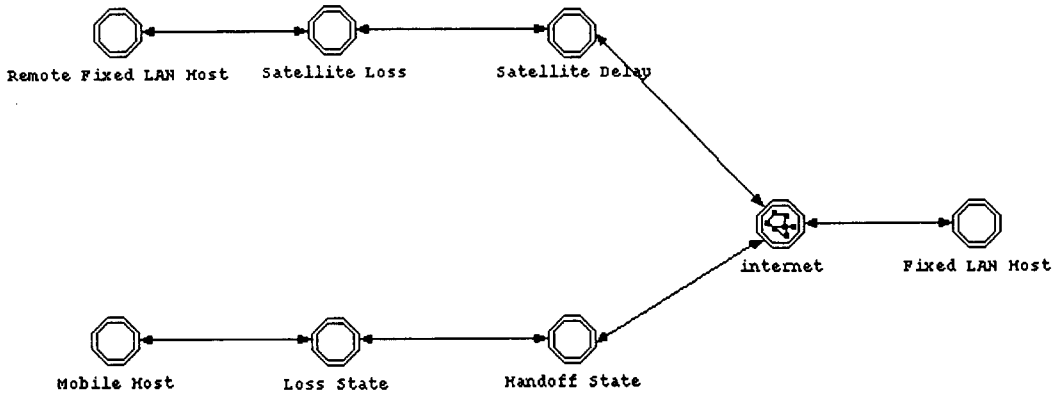

Figure B.1. The heterogenous wireless and wired simulation envrionment model

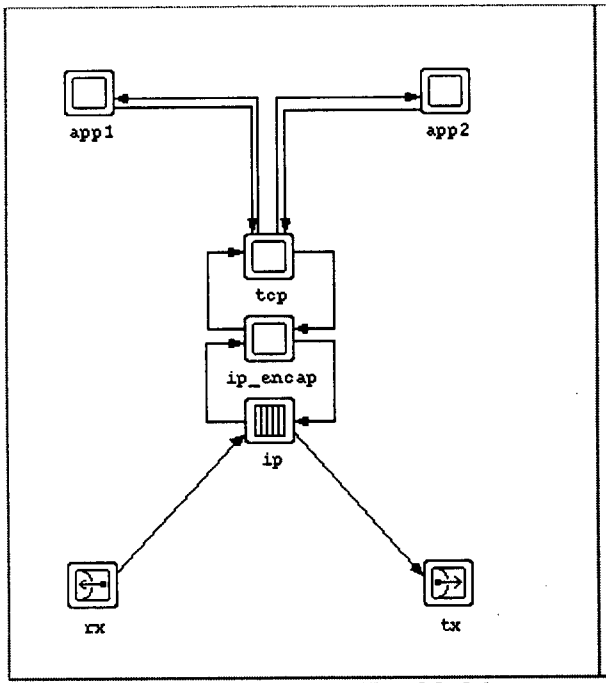

Figure B.2 A End Host Model

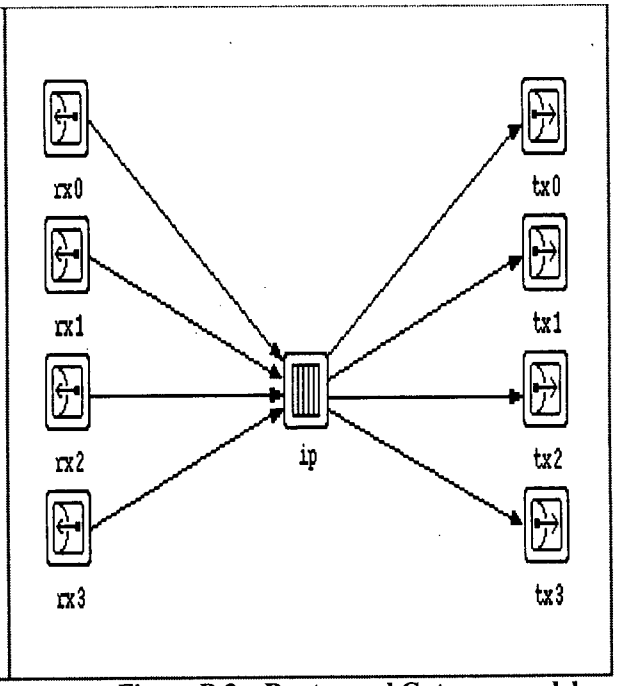

Figure B.3 Router and Gateway model

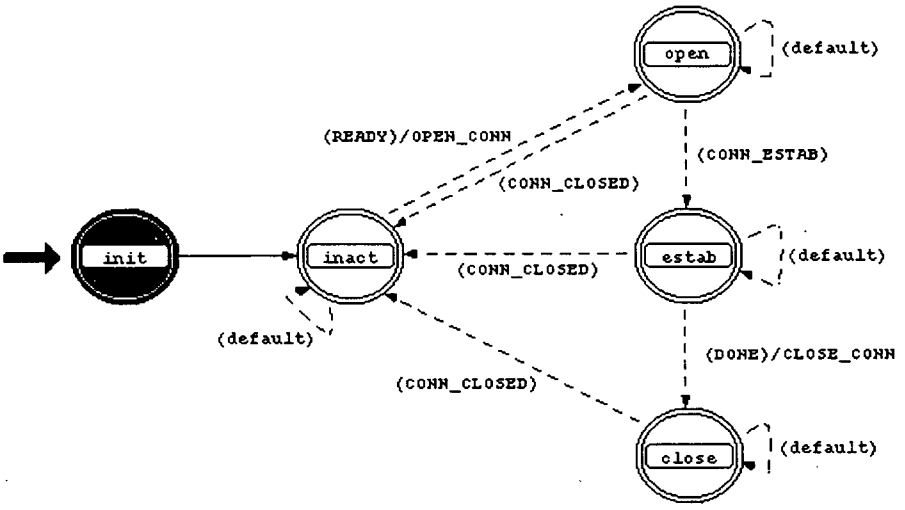

Figure B.4 The TCP application process model

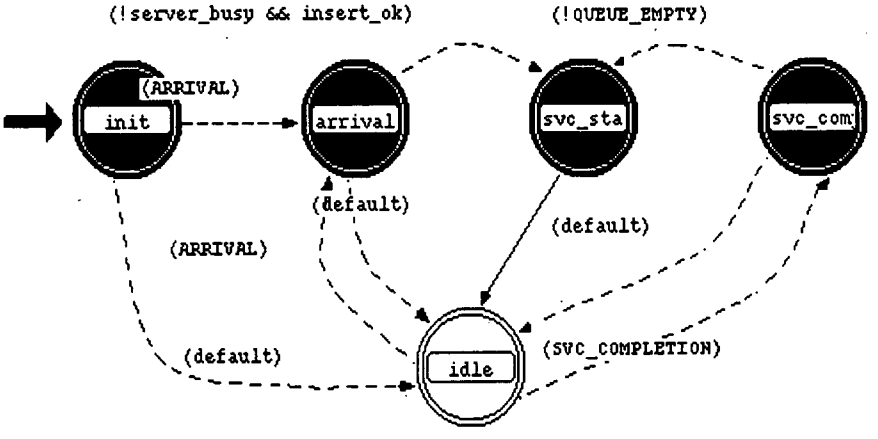

Figure B.5 The IP and state process models