DIFF-C-TCP: A NEW TCP USING ECN MARKS TO IMPROVE THE TCP PERFORMANCE OVER WIRELESS LINKS

Thesis

Submitted to

The School of Engineering of the

UNIVERSITY OF DAYTON

in Partial Fulfillment of the Requirements for

The Degree

Master of Science in Electrical and Computer Engineering

by

Haowei Bai

UNIVERSITY OF DAYTON

Dayton, Ohio

August, 2001

DIFF-C-TCP: A NEW TCP USING ECN MARKS TO IMPROVE THE TCP PERFORMANCE OVER WIRELESS LINKS

APPROVED BY:

Mohammed Atiquzzaman, Ph.D.
Advisory Committee Chairman
Associate Professor, School of Computer Science
University of Oklahoma

Majeed Hayat, Ph.D.
Committee Member
Associate Professor, Electrical and Computer Engineering Department

Russell C. Hardie, Ph.D.
Committee Member
Associate Professor, Electrical and Computer Engineering Department

Donald L. Moon, Ph.D.
Associate Dean
Graduate Engineering Program & Research
School of Engineering

Blake Cherrington, Ph.D., P.E.
Dean, School of Engineering

ABSTRACT

DIFF-C-TCP: A NEW TCP USING ECN MARKS TO IMPROVE THE TCP PERFORMANCE OVER WIRELESS LINKS

Name: Bai, Haowei

University of Dayton, 2001

Advisor: Dr. Mohammed Atiquzzaman

TCP was designed for wireline networks, where loss events are mostly caused by buffer overflows. The congestion control mechanism of current TCP uses loss events as the congestion indication which results in TCP reducing its congestion window size. However, when a wireless link is involved into the TCP connection, TCP performs poorly which is due to the characteristics of wireless links that are fundamentally different from wired links, especially the loss behavior. The congestion window size should not be decreased if the loss event is caused by link corruption. In this thesis, for better understanding of characteristics of wireless links, we first present a novel approach to classify the existing wireless error modeling methods, based on different requirements of different researchers in the practical engineering; we then set up a complete model to show that zero congestion loss could be achieved by appropriately setting the ECN (Explicit Congestion Notification) marking threshold. Based on

the understanding of wireless links and the possibility of zero congestion loss, we propose a new TCP algorithm, called Differentiation Capable TCP (Diff-C-TCP), which assumes packet losses to be the indicator of link corruption and uses ECN as the indicator of network congestion to differentiate between congestion and corruption losses over lossy links. The *main contribution* of this thesis is that a complete model is set up for achieving zero loss of TCP congestion control by appropriately choosing the RED threshold and buffer size; and then the Diff-C-TCP is proposed to improve the TCP performance in the lossy environment. We have shown that the proposed algorithm performs very well in the presence of lossy links having significant amount of corruption losses in addition to congestion losses.

ACKNOWLEDGMENTS

My special thanks are in order to Dr. Mohammed Atiquzzaman, my academic advisor, for directing this thesis and my other graduate research with patience and expertise. I would never forget his encouragement and enlightening conversations during my study and research.

I would also like to express my appreciation to my committee members, Dr. Majeed Hayat and Dr. Russell C. Hardie for their help and discussions on this work and my study, and for their time for reviewing this thesis.

I also want to thank many people who have helped me with my graduate study. This includes Dr. Donald L. Moon, Dr. Partha Banerjee, Dr. Tommy Williamson, Ms. Karen Hardie, Ms. Joann B. Catanzaro and Mr. John A. Fortune.

Most importantly, I would like to express my appreciation to my grandparents, my mother and my father. All work would not have been possible without their encouragement and support.

Contents

List of Figures

List of Tables

Chapter 1

INTRODUCTION

The TCP/IP (Transmission Control Protocol/Internet Protocol) protocol for the Internet was originally designed for wireline networks. Wireless links have a few fundamentally different characteristics from wired links. They are dominated by low bandwidth and high error rates, which degrade the performance of TCP. In the current TCP, the congestion control mechanism uses packet loss as the congestion indication which result in TCP reducing its congestion window size. However, when the packet loss is caused by corruption rather than congestion, TCP does not need to reduce the congestion window size. Unfortunately, packet losses due to corruption are more significant than congestion losses when a lossy link is involved in a TCP connection. In such a case, TCP may not be able to transmit or receive at the full available bandwidth, because the TCP algorithm is spending time in slow-start or congestion avoidance procedures triggered by link errors [1]. Consequently, the cur-

rent congestion control algorithms in TCP result in very poor performance over lossy links.

1.1 TCP Extensions for Wireless Links

Several schemes have been proposed to improve performance of TCP over wireless links. These can be classified into two classes. In first approach, the TCP sender is unaware of the losses due to wireless link. So the TCP at the sender does not need to be changed. In second approach, the sender is aware of the existence of the wireless link in the network and attempts to distinguish the losses due to wireless link from that due to congestion. So the sender does not invoke congestion control algorithms when the data loss is due to wireless links [3]. We describe the proposed solutions in this section.

1.1.1 Wireless Aware TCP

In this approach, the fixed host (sender) is aware of the existence of the wireless link in the network and is able to distinguish the losses due to transmission error on wireless link from those due to congestion. The sender can avoid invoking congestion control algorithms when the losses are due to the wireless link. We now discuss TCP extensions, which are based on this approach.

Slow-Start and Congestion Avoidance

Recently, TCP has been improved to provide more efficiency and reliability in the case of packet losses over large networks. The Slow-Start algorithm [4, 5] is used to avoid inappropriately transmitting a large amount of traffic following a packet loss. It forces TCP to transmit one segment and wait for the corresponding ACK (Acknowledgement). During a slow start period, the value of *cwnd* (Congestion Window) is increased by one with each ACK received by the sender. When the *cwnd* size is greater than or equal to *ssthresh* (Slow Start Threshold), the Congestion Avoidance algorithm [4, 5, 6] is used so that the *cwnd* increases slower than during slow start.

Fast Retransmit and Fast Recovery

Above two algorithms lead to poor utilization of the available channel bandwidth when they are used in long-delay satellite networks [7]. Because the Slow-Start and Congestion Avoidance mechanisms are generally considered to be essential to well-behaved TCP implementations on the Internet, those two mechanisms appear to have a limiting effect on TCP's performance over lossy links [8]. Moreover, they prevent TCP from quickly recovering from packet losses. Fast Retransmit and Fast Recovery algorithms [6, 9] are used to reduce the *cwnd* by half before transmitting new data. However, Fast Retransmit and Fast Recovery algorithms can lead to a problem that allows multiple fast retransmits per window of data [10], resulting in the congestion window size being reduced multiple times in response to a single loss event. This may

hurt the performance of TCP [11], especially over corruption-dominated lossy links.

Selective Acknowledgements

The authors in [12] describe a conservative extension to the Fast Recovery algorithm by taking into account the information provided by Selective Acknowledgements (SACKs) [13]. When SACK based algorithms are used, the sender can be exactly informed which packets need to be retransmitted in the first RTT (Round Trip Time) following the loss event. In this way, SACK allows TCP to recover from multiple segment losses in a window of data within one RTT of loss detection [14]. Although Fast Retransmit, Fast Recovery and SACK are generally able to rapidly recover from multiple packet losses, they reduce the congestion window to avoid further congestion. The above behavior, which is based on the assumption that packet losses are indicators of congestion, results in a degradation of throughput in the presence of non-congestion related packet losses (such as wireless link errors). Therefore, when they are applied to lossy links, where most of packet losses are due to link errors instead of congestion, TCP is unable to determine the available bandwidth.

Using Heuristic Method to Differentiate Congestion losses and Corruption Losses

Some researchers applied heuristics, such as loss predictors, to distinguish between congestion and transmission errors [15, 16], but their simulation measurements in-

dicated that their loss predictors did not perform well. They proposed a modified TCP-Reno scheme called TCP-Aware [17] to differentiate between packet losses due to congestion and losses due to link errors. However, their scheme works well only when the last hop for the connection is wireless, the bandwidth of the wireless link is much smaller than the bandwidth of the wired link, and the overall packet loss rate is small [17], all of which are too critical and limited in the real world Internet.

Distinguishing Losses by Making Two Connections

The basic model for this method is that the network is wired and the last hop from the base station to the wireless host is wireless. It is assumed that the wireless host receives all its packets from the base station. This method does not take care of the handoffs. The TCP at the sender is assumed to know that the destination host is a wireless host.

The key idea in this algorithm is that the sender distinguishes between congestion on the wired network and the transmission errors on the wireless part. The wireless host assures that if the packets are dropped due to transmission error, the sender retransmits them before its timeout [18]. When a fixed host wants to communicate with a wireless host, it opens two connections, one with the base station and the other with the wireless host. The connection between the sender and the base station is called control connection. This is used to estimate the congestion on the wired link. The packets on these two connections are expected to be routed in the same

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

1.1.2 Wireless Unaware TCP

In this approach, the non-congestion related losses are hidden from the TCP at the fixed host (sender), and hence the TCP at the fixed host remains unmodified. This approach is based on the intuition that since the problem is local, it should be solved locally and the TCP should be independent of the behavior of the individual links. We now present some solutions based on this approach.

The Snoop Module

A simple protocol called *snoop* protocol that improved TCP performance over wireless network has been proposed in [3]. The main idea of their protocol is to cache packets at the base station and perform local retransmissions over wireless links. However, this protocol assumes that the wireless link is the last hop in the TCP connection. It also needs a base station to maintain the state information, and cache the unacknowledged TCP packets, which results in an increased requirement for resources.

I-TCP (Indirect TCP)

This was implemented at Rutgers University as a part of the Dataman Project. The scheme works by breaking the connection between the machine on the fixed wired network and the wireless mobile host in two connections. One connection is between the fixed host and the base station; the other connection is between the base station and the wireless host. Data sent to the wireless host is first received by the base station. Upon receiving the data, the base station sends an acknowledgement to the fixed host and then the received data is forwarded to the wireless host. The base station and the wireless host does not need to use TCP for communication. Instead a specialized protocol that is optimized for mobile applications and for low speed and unreliable wireless medium can be used.

This indirection helps shield the wired network from the uncertainties of the wireless network. The TCP/IP at the fixed host side does not to be changed. Because

of this indirection, the wireless host could be very simple and the base station would handle most of the complexity about communication overhead. If the wireless host moves to a different cell while communicating with a fixed host, the whole connection information is transferred from the current base station to the new base station and the new base station takes over from here. The fixed host is unaware of this indirection and is not affected when this switch occurs.

However, I-TCP violates the acknowledgement mechanism of current TCP, because acknowledgements of data packets would possibly reach the original source before the data packets reach the wireless host. If errors due to link corruption happens, since acknowledgements could be received by the original sender before the wireless host's receiving data packets, it would be dangerous.

1.2 Zero Loss Congestion Control with Explicit Congestion Notification

As seen from the description in Section 1.1, if losses due to congestion and corruption in wireless links could be appropriately differentiated, significant performance improvements can be achieved [2]. Without any other additional information, the existing implicit loss feedback mechanisms in TCP does not allow distinguishing between congestion and corruption losses [1]. ECN (Explicit Congestion Notification) [19] was proposed as an explicit indicator of congestion. It can be used to quickly and unam-

biguously inform sources of network congestion, without the sources having to wait for either a retransmit timer timeout or three duplicate ACKs (Acknowledgements) to infer a dropped packet. For bulk-data connections, this mechanism can avoid unnecessary packet drops for low-bandwidth delay-sensitive TCP connections, and can avoid some unnecessary retransmit timeouts in TCP [20]. Since ECN is an explicit indicator of network congestion, it provides the possibility to differentiate between losses due to congestion and losses due to link errors. If the buffer has enough space and the threshold of the RED (Random Early Detection) router is appropriately set, zero-loss congestion control could be achieved by appropriately adjusting the source's window size based on the notification of ECN signal. Some researchers have done some initial work.

Authors in [21] presented a framework for designing end-to-end congestion control schemes in a network where each user may have a different utility function. They considered ECN marks as an alternative of losses for the congestion notification. Using this model, they showed that the ECN marking level can be designed to nearly eliminate congestion losses in the network by choosing the marking level independently for each node in the network. However, they ended up with an apparent conclusion that increasing the network resource is the only option to ensure loss-free service.

Authors in [22] analyzed the queue dynamics at the congested router, and derived the closed-form formula and buffer requirements to achieve zero loss and full link utilization. However, as they stated in their work, they didn't get the mathematical

expression for the average share of bottleneck link bandwidth which is the basis of their model. Instead, they used the simulation to illustrate the relationship between the average share of bottleneck link bandwidth and the Round Trip Time (RTT). The same difficulty is also encountered by authors in [23]. As a solution, they introduced an unknown constant into the final expression.

1.3 Our Objective

Motivated by the above results, in this thesis, we model the ECN mechanism and deduce the zero congestion loss requirements. As one of the most important parameters in our model, we also derive the exact mathematical expression for the average share of bottleneck link bandwidth by modeling the ECN marking dynamics as a Poisson Process. We finally end up with a complete mathematical model for achieving the zero-loss TCP congestion control.

In the heterogeneous network environment involving lossy links, if we can eliminate all network congestion losses, or if the congestion losses are a small fraction of random losses, with the negligible error all losses can be attributed to random losses. This finally leads to our proposed Diff-C-TCP discussed in Section 5.2. Diff-C-TCP assumes loss events indicate link corruption and uses ECN as the congestion indication with the precondition of zero congestion losses to differentiate between congestion and corruption. As mentioned earlier, because of links errors, packet loss due to corruption is more significant in a lossy network. To have a high TCP throughput

when a TCP connection traverses a lossy link, the TCP source should persist in the previous utilization of bandwidth instead of reducing the transmission rate when the loss is due to corruption.

The *contributions* of this thesis are:

- For better understanding of the characteristics of wireless links, a novel approach to classify the existing wireless error modeling methods is presented, based on different requirements of different researchers in the practical engineering.

- A complete analytic model is set up for achieving zero congestion loss with the use of ECN mechanism; the exact mathematical expression of average share of bottleneck link bandwidth with multiple competing TCP flows is deduced.

- Finally, based on the zero congestion loss model and the understanding of the characteristics of wireless links , the Diff-C-TCP is proposed to improve the performance of TCP over lossy links.

The *significance* of our proposed algorithm is that it can deal with multiple packet losses that are due to link errors without reducing the congestion window size, thereby resulting in a high throughput in the presence of corruption losses. At the same time, it appropriately reduces the congestion window size in the presence of network congestion, thereby helping the network to get out of congestion.

This thesis is organized as follows. In Chapter 2, for better understanding of the characteristics of wireless links, we present a novel approach to classify the existing

wireless error modeling methods. In Chapter 3, we briefly describe the loss detection schemes deployed by current TCP. In Chapter 4, we set up a complete model used to achieve zero congestion losses with multiple competing TCP flows which is the basis of our proposed Diff-C-TCP algorithm discussed in Chapter 5. In Chapter 6, we describe the simulation methodology that has been used to test the performance of our proposed algorithm. Performance improvements achieved by our proposed algorithm are also shown in this chapter. Concluding remarks are finally given in Chapter 7.

Chapter 2

PROPAGATION CHARACTERISTICS AND ERROR MODELING FOR WIRELESS COMMUNICATION CHANNELS

2.1 Introduction

Several physical media, ranging from light to radio, could be used for wireless communication, which leads to the complexity of modeling wireless channels, because

of several reasons. First, wireless channels are inherently dependent on the physical properties of transmission media, such as reflections from a smooth surface, diffractions around a corner and scattering caused by a lamp post. Secondly, the characteristics of wireless channels are closely related to the characteristics of different geometric sites in which the wireless channel is formed. It is known that the mean power is the basic requirement for reliable wireless communications. The energy should be sufficient, but not too strong, in order to avoid cochannel interference. Since the radio link is highly variable over short distances due to the statistical distribution of PL (Path Loss) and physical properties of propagation environments, the statistical distribution of channels is also very important. From another point of view, even if there is adequate power for communication, the quality of received signal might not be perfect. This arises from rapid movement through the scattering environment, or impairments due to long echoes leading to inter-symbol-interference [24]. Again, it is necessary to obtain a basic understanding of wireless channels in order to find better modulation and coding schemes to improve the quality of channels.

Several effects that can lead to bit errors and packet losses over a wireless channel are described below.

- The quality of received signal is often evaluated by *Signal-to-Noise Ratio* (SNR). If the SNR value is too low, the received signal will not be detected at the receiver, consequently yielding bit errors and packet losses. However, note that SNR is not the only factor leading to errors.

- The second one is *Inter-Symbol-Interference* caused by delay spread, which means that the arrival of an earlier transmitted symbol is delayed, thereby partially cancelling out the current symbol.

- The third one is *Doppler Shift* due to relative velocities of the transmitter and the receiver. Doppler shift causes frequency shifts in the arrived signal, thereby complicating the successful reception of the signal.

It is difficult for researchers, especially for wireless network protocol developers, to consider so many factors that affect the error performance of wireless channels. Therefore, wireless error models will be very helpful in evaluating the performance of wireless networks and communication systems.

Wireless channels are usually modeled by capturing the statistical nature of the interaction among reflected radio waves. The statistical calculations for *Bit Error Rate* (BER), which is usually used to characterize channel errors at the physical layer, is a well-known practice. From the perspective of higher layer, network protocol developers and algorithm designers are interested in block errors (packet errors), since almost all of the applications running on top of link layers exchange blocks of data. It has been observed that wireless errors can be approximated using a two-state Markov process in order to more easily evaluate the block-error rate. Both statistical calculations for BER at the physical layer and error approximation at higher layers use *analytical modeling methods* which are described in this survey. In addition to analytical modeling methods mentioned above, it is highly desirable that wireless

networks be modeled in a thoroughly repeatable fashion, which is especially necessary for people who must experiment with realistic channel parameters. The results from analytical modeling methods are either inaccurate or unlikely to be reproduced, giving rise to another modeling method named *empirical distribution-based modeling methods* in this survey. The empirical distribution-based methods are capable of solving this problem.

In this chapter, our *objective* is to present a novel approach to classify the existing wireless error modeling methods. In contrast to traditional classification methods, our *approach* is based on different requirements of different researchers in the practical engineering. Our *goal* is to provide the appropriate method for modeling wireless error channels to different researchers. According to our classification, *we recommend* that researchers who need to analyze characteristics of wireless error channels should use *Analytical Models*; network protocol designers and evaluators who need to synthesize a wireless error channel, i.e., build an appropriate error model for their simulations and evaluations over special communication environments, should use *Empirical Distribution-Based Models.*

The rest of this chapter is organized as follows. The nature of radio propagation is described in Section 2.2 followed by discussion of the characteristics and models of fading channels in Section 2.3. Several wireless error modeling methods are classified and introduced in Section 2.4. Finally, concluding remarks are given in Section 2.5.

2.2 Properties of Radio Propagation

There are three main mechanisms that impact radio propagation in a wireless communication environment [25] as described below:

- *Reflection*, which may interfere constructively or destructively at a receiver, occurs when a propagating electromagnetic wave impinges on a smooth surface with very large dimensions compared to the wavelength of the radio wave.

- *Diffraction* occurs when the radio path between the receiver and the transmitter is obstructed by an impenetrable body with large dimensions as compared to the RF (radio frequency) signal wavelength. This causes secondary waves to be formed behind the obstructing body, even though there is no LOS (line of sight) path between the two. Diffraction, which is also named *shadowing* because the diffracted field can reach the receiver even when shadowed by an impenetrable obstruction, explains how RF energy can travel in urban and rural environments without a LOS path.

- *Scattering* occurs when the radio channel contains objects with dimensions that are on the order of the wavelength or less, causing energy from a transmitter to be radiated in many different directions. In urban environments, typical examples that yield scattering are lamp posts, street signs and foliage.

2.3 Fading Channels

In a practical wireless communication system, a signal may travel between the transmitter and the receiver over multiple paths, which is referred to as *multipath* propagation. Multipath propagation can cause fluctuations in the received signal's amplitude, phase and angle of arrival, giving rise to the term *multipath fading*. In this section, we describe the various types of fading followed by models of fading channels.

2.3.1 Large-scale Fading and Small-scale-Fading

Based on the distance over which the mobile moves, there are two different types of fading effects: large-scale fading and small scale fading [26]. If the mobile moves away from the transmitter over a large distance, the received signal will experience large-scale signal variation. *Large-scale fading* represents the average signal power attenuation and path loss due to motion over large areas. The receiver is often represented as being shadowed by such prominent terrains as hills, forests, billboards, clumps of buildings, etc.

Small-scale fading refers to the dramatic changes in signal amplitude and phase that can be experienced as a result of small changes (as small as a half-wavelength) in the spatial separation between the transmitter and the receiver. When there are a large number of reflective paths and there are no LOS signal components, the envelope of the received signal can be statistically described by the Rayleigh distribution [27] [28]. If dominant non-fading components exist, such as LOS propagation path, the

small-scale fading envelope is Rice distributed [27] [28]. A mobile radio propagating over a large area will experience both types of fading, in other words, small-scale fading super-imposed on large-scale fading.

2.3.2 Modeling of Fading Channels

In this section, we present the different types of fading channels typical of practical communication environments and the mathematical models that can be used to describe the channels. Table 2.1 [29] shows these various fading channel models classified by environments to which they apply.

Multipath fading arises from the constructive and destructive combination of randomly delayed reflected, scattered, diffracted signal components. Based on the nature of the radio propagation environment, there are different mathematical models describing the statistical behavior of the multipath fading envelope.

- The *Rayleigh* distribution typically agree very well with experimental data for mobile system where no LOS path exists between the transmitter and receiver antennas [30].

- The *Nakagami-q* distribution is typically observed on satellite links subject to strong ionospheric scintillation [31].

- The *Nakagami-n* distribution, known as the *Rice* distribution, is often used to model channels consisting of one strong direct LOS component and many

random weaker components [32].

- The *Nakagami-m* distribution often gives the best fit to land-mobile, indoor-mobile multipath propagation, as well as scintillating ionospheric radio links [33].

In terrestrial and satellite land-mobile systems, due to the effects of shadowing from terrain, buildings and trees, the link quality is also affected by slow variation of the mean signal level. The shadowing can generally be modeled by a *log-normal* distribution for various outdoor and indoor environments [34]. If the receiver is able to average out the fast multipath fading, the performance of mobile system depends only on *shadowing*. However, in the environment consisting of multipath fading superimposed on shadowing, the receiver does not average out the fading envelope. This scenario, named *composite multipath/shadowing*, is typically observed in congested downtown areas with slow moving pedestrians and vehicles [35]. A detailed discussion of this topic and corresponding probability density functions of fading amplitude and SNR (the signal-to-noise ratio) can be found in [36].

2.4 Error Models of Fading Channels

With the increasing deployment of wireless networks, most of research has focused on improving the quality of wireless systems. This has led to a growing interest in characterizing the loss behavior of many wireless technologies. A number of error models have been proposed in the literature to simulate the loss behavior of transmission

channels. The errors models can be classified into two groups: *analytical models* and *empirical distribution based models.*

2.4.1 Analytical Models

Errors occurring on wireless channels are due to the diversity of wireless connections and the complicated physical impairments. As a result, it is difficult to generalize the mathematical results from one specific domain to another. Therefore, analytical modeling methods are highly dependent on the characterization of error environments, such as fading channels.

The traditional metric used for characterizing channel errors at the physical layer is the average BER (bit-error-rate), which gives rise to the first class of analytical modeling methods called *physical layer oriented* modeling which deals with bit errors. It is expected that future wireless communications will include image, video and data applications, which requires the efficient usage of spectral resources. In order to take this into effect, the mixed media encoding algorithms will have to carefully designed to overcome the impact of wireless errors.

Though wireless channel errors are due to various physical impairments, the channel interactive with those higher layer applications is never the raw physical channel. At the same time, higher layer applications running on top of a link layer usually manage data transfer in blocks containing multiple bits or symbols and employ various block error detection and retransmission mechanisms. This gives rise to the second

class of analytical modeling methods named *higher layer oriented* modeling which is interested in block errors. In the following subsections, we describe both physical layer oriented modeling and higher layer oriented modeling.

Physical Layer Oriented Modeling

This method aims at determining the BER performance of a wireless communication system over fading channels. Based on the specific channel model and modulation/detection combination, the average BER is obtained through statistical calculations. A lot of results have been reported using this modeling method [45] [46] [47] [48] [49] [50] [51] [52]. They deal with different fading channels and combinations of modulation/detection techniques. However, a useful generalized expression [29] for the average BER performance of wireless communication systems over AWGN (Additive White Gaussian Noise) and fading channels which unifies all the results mentioned above is given in Equation 2.1.

Figure 2.1 [29] shows a typical multilink channel model, which is used to develop the generalized model for determination of the BER performance. As shown in Figure 2.1, the transmitted signal is received over L independent channels. Each of them is a fading channel. In Figure 2.1, $\{r_l(t)\}_{l=1}^{L}$ is a set of L received replicas of the signal, where l is the index of the channel, and α, θ, τ are the random fading channel amplitudes, phase and delays, respectively. The first channel (with index of 1) is assumed to be the reference channel whose delay is 0. The fading amplitude, α_l of

the lth channel is a random variable with a mean squared value of $\overline{\alpha_l^2}$ which is denoted by Ω_l. The probability distribution of the random variable α_l is any of the family of distributions presented in Table 2.1. Based on this channel model, authors in [29] use alternate representations of Gaussian and Marcum Q-functions that are characteristic of error-probability expression for coherent, differentially coherent and noncoherent forms of detection to obtain the generalized BER expression as follows:

$$\begin{aligned} P_b(E;\gamma) &= Q_1(a\sqrt{\gamma}, b\sqrt{\gamma}) - \left[1 - \frac{\sum_{l=0}^{L_r-1}\binom{2L_r-1}{l}\eta^l}{(1+\eta)^{2L_r-1}}\right] \times exp\left[-\frac{(a^2+b^2)\gamma}{2}\right] I_0(ab\gamma) \\ &+ \frac{1}{(1+\eta)^{2L_r-1}} \times \left[\sum_{l=2}^{L_r}\binom{2L_r-1}{L_r-1}\eta^{L_r-1} \times [Q_l(a\sqrt{\gamma}, b\sqrt{\gamma}) - Q_1(a\sqrt{\gamma}, b\sqrt{\gamma})]\right. \\ &- \left.\sum_{l=2}^{L_r}\binom{2L_r-1}{L_r-l}\eta^{L_r-1+l} \times [Q_l(b\sqrt{\gamma}, a\sqrt{\gamma}) - Q_1(b\sqrt{\gamma}, a\sqrt{\gamma})]\right], \end{aligned} \tag{2.1}$$

where the function $Q(\cdot)$ is the generalized Marcum Q-function obtained by the authors in [29] and is given by

$$Q_l(\alpha, \beta) = \frac{1}{\alpha^{l-1}} \int_\beta^\infty x^l exp\left[-\left(\frac{x^2+\alpha^2}{2}\right)\right] I_{l-1}(\alpha x) dx. \tag{2.2}$$

The $I(\cdot)$ function in Equation (2.2) is the first kind $(l-1)$st order modified Bessel function. The parameter l in Equation (2.1) is the channel index, $\gamma = \sum_{l=1}^{L_r}\gamma_l$ is the total instantaneous SNR per bit.

Typical values of η, a and b corresponding to specific modulation/detection schemes are shown in Table 2.2 [29]. Note that in all possible cases, a and b are independent of

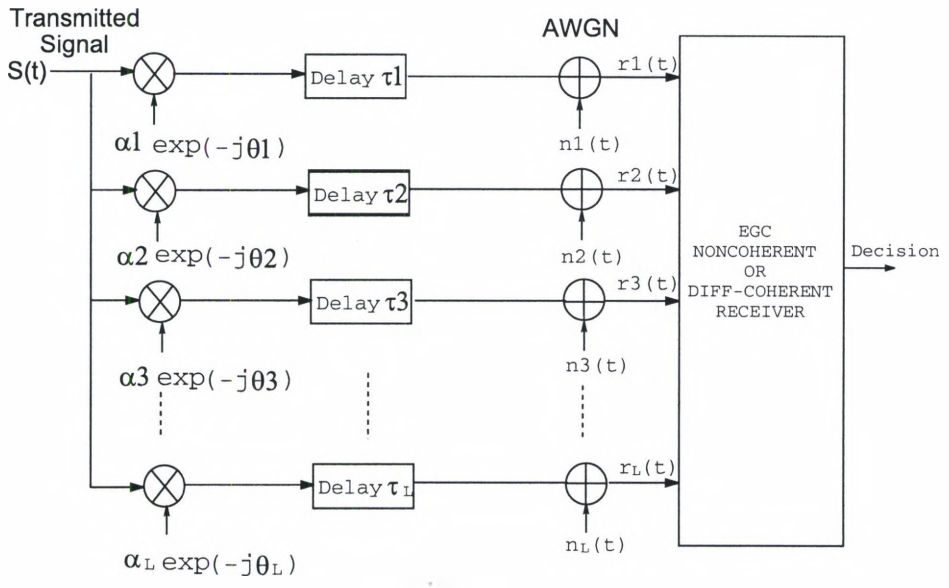

Figure 2.1: Generalized fading channel

the fading channel model. For the case of single channel reception, the value of L_r in Equation (2.1) is equal to one. Otherwise, $L_r \geq 1$, which corresponds to the case of multichannel detection. Details on the derivation of this generalized expression can be found in [29].

Example: The usefulness of the generalized BER equation (as given by 2.1) is shown by an example below, in which, by plugging in appropriate parameters summarized in Table 2.2, the well-known conditional BER expression for orthogonal DPSK with single channel reception is obtained.

Let $L_r = 1$ (i.e., single channel reception). The latter two summations in Equation

(2.1) do not contribute. One can immediately obtains the result, i.e.,

$$P_b(E;\gamma) = Q_1(a\sqrt{\gamma}, b\sqrt{\gamma}) - \left(\frac{\eta}{1+\eta}\right) exp\left[-\frac{(a^2+b^2)\gamma}{2}\right] I_0(ab\gamma). \tag{2.3}$$

For η=1, a=0 and b=$\sqrt{2}$ (See Table 2.2), Equation (2.3) can be reduced to the well-known expressions for DPSK as reported by a lot of authors, which is given by

$$P_b(E;\gamma)|_{DPSK} = \frac{1}{2} exp(-\gamma). \tag{2.4}$$

Higher Layer Oriented Modeling

As mentioned in Section 1, modeling of higher layers aims at calculating the average block-error rate (packet-error rate) which is very important to the performance analysis of link layer protocols. The special structure of Markov models makes them naturally useful and tractable [53]. The study of Markov approximation dates back to early work of Gilbert [54] and Elliott [55]. They built a two-state Markov channel known as the Gilbert-Elliott channel. In a simplified Gilbert model [56], the error probabilities in bad state and good state are 1 and 0, respectively. Assume 0 and 1 denote successful and erroneous transmission in a given slot, respectively, and let

$$\mathbf{P} = \begin{pmatrix} P_{00} & p_{01} \\ p_{10} & p_{11} \end{pmatrix} \tag{2.5}$$

be the transition matrix for the packet-error process. The average packet-error rate is then given by [57]

$$\varepsilon = \frac{p_{01}}{p_{10} + p_{01}}. \tag{2.6}$$

This Markovian model for average packet error probability can be easily extended for diversity. The performance can be improved by using two (or more) suitable spaced antennas over fading channels [57]. For example, if a diversity of order two (i.e., two antennas) is employed, and the signals received at the two antennas fade independently, the channel can be modeled by three states: both channels are good (state 0), two channels have different states (state 1), and both channels are bad (state 2). If the transition matrix of each channel has the same form as Equation (2.5), the transition matrix of this three-state Markov process can be written as [57]

$$\mathbf{P} = \begin{pmatrix} p_{00}^2 & 2p_{01}p_{00} & p_{01}^2 \\ p_{10}p_{00} & p_{11}p_{00} + p_{10}p_{01} & p_{01}p_{11} \\ p_{10}^2 & 2p_{10}p_{11} & p_{11}^2 \end{pmatrix}. \tag{2.7}$$

In this model, both state 0 and state 1 correspond to successful transmission, while state 2 corresponds to a transmission failure.

A Binary Symmetric Channel (BSC) with a given crossover probability can be associated with each state so that the channel quality for each state can be identified. Unfortunately, the Gilbert-Elliott channel is a special case, because the crossover

probability of the BSC are 0 and 0.5, respectively [58]. When the channel quality varies dramatically, a two-state Gilbert-Elliott model is not adequate. As a solution, a FSMC (finite-state Markov channel) is proposed in [58] as the extension of two-state Markov model. By partitioning the range of the received SNR into a finite number of intervals, FSMC models can be constructed for Rayleigh fading channels [58].

2.4.2 Empirical Distribution-Based Models

A large number of measurements have been conducted by telecommunication companies, research laboratories, and universities in order to determine reasonable propagation parameters for wireless communication systems. These measurements show that environmental factors, such as terrains, construction materials, speed of pedestrians and vehicles, etc., have a direct impact on radio propagation characteristics. Therefore, although analytical modeling methods as described are sound in theory, in practice it is hard to determine the values of the parameters in these models, especially when we want to build a specific model under a special environment. This gives rise to *empirical distribution-based* modeling methods.

Empirical distribution-based models consist of three phases: *Data collection*, *statistical analysis and model construction*, *model validation*.

- In the *data collection* phase, a large number of statistical data are collected and recorded by network tracing or measurements for many different scenarios.

- Those statistical data are extracted in terms of interests, such as packet errors,

and are modeled in the *statistical analysis and model construction* phase.

- The model is built by fitting known probability distributions to data. In the *model validation* phase, the models are simulated or analyzed and validated. The models are refined to make them consistent with the collected data and traces.

Example: A good example of an empirical distribution-based model is presented in [59] where the loss behavior of the AT&T WaveLAN, a popular in-building wireless interface, is characterized and modeled using empirical distribution-based modeling method. Packet loss information is recorded and analyzed by conducting network tracing. As a result, evaluating wireless network protocols with a uniform error model has been shown to be inaccurate [59]. Furthermore, authors in [59] also reveal that the wireless error behavior of WaveLAN can not be accuratley modeled by a simple two-state Markov chain using analytical modeling method. Instead, another improved two-state Markov model, based on the distribution of the error length (defined to be the number of packets that are lost consecutively) and error-free length (defined to be the number of packets that are successfully received between two adjacent bursts of errors [59]) of the packet stream has been shown to be more appropriate. The key difference between the two lies in the probability distribution of the error and error-free length. The simple two-state Markov model assumes that the error length and error-free length are *geometrically* distributed. However, results of the empirical distribution-based modeling shows that the error length is better described

by a combination of two *Exponential* distributed segments, and the error-free length is better described by a combination of three segments, two *Pareto* distributions and one *Exponential* distribution. The above example shows the usefulness of the empirical distribution-based modeling method for modeling wireless errors.

2.5 Summary

Wireless error modeling methods have been introduced in this chapter. A number of error models used to approximate the loss behavior of transmission channels have been classified into two groups: *analytical models* and *empirical distribution-based models.* Since errors occurring on wireless channels are due to the diversity of wireless connections and the complicated error environment, analytical modeling methods are highly depended on the characterization of error environments, such as fading channels. Although physical layer oriented modeling methods and higher layer oriented modeling methods can be used to evaluate the BER performance and packet error performance respectively, it is highly desirable that wireless errors be modeled in a thoroughly reproducible way, which is especially necessary for developers of network protocol and mobility algorithms who must experiment with realistic channel parameters. The empirical distribution-based modeling approach which alleviates the above problem has been described in this article.

According to our classification, *we recommend* that researchers who need to analyze characteristics of wireless error channels should use *Analytical Models*; network

protocol designers and evaluators who need to synthesize a wireless error channel, i.e., build an appropriate error model for their simulations and evaluations over special communication environments, should use *Empirical Distribution-Based Models.*

Table 2.1: Models that can be used to characterize various wireless environments.

Environment	*Channel Type*
Mobile systems with no LOS path between transmitter and receiver antenna, propagation of reflected and refracted paths through troposphere and ionosphere, ship-to-ship radio links [37].	Rayleigh [30]
Satellite links subject to strong ionospheric scintillation [38].	Nakagami-q (Hoyt) (spans range from one-sided Gaussian (q=0) to Rayleigh (q=1)) [31]
Propagation paths consisting of one strong direct LOS component and many random weaker components - microcellular urban and suburban land mobile, picocellular indoor and factory environments [39].	Nakagami-n (Rice) (spans range from Rayleigh (n=0) to no fading (n=∞)) [32]
Often best fit to land mobile [40], indoor mobile multipath propagation as well as ionospheric radio links.	Nakagami-m (spans range from one-sided Gaussian (m=$\frac{1}{2}$), Rayleigh (m=1) to no fading (m=∞)) [33]
Caused by terrain, buildings, trees - urban land mobile systems, land mobile satellite systems [42].	Log-Normal shadowing [41]
Nakagami-m multipath fading superimposed on log-normal shadowing. Congested down town areas with slow-moving pedestrians and vehicles. Also in land mobile systems subject to vegetative and/or urban shadowing [43].	Composite gamma/log-normal [41]
Convex combination of unshadowed multipath and a composite multipath/shadowed fading. Land mobile satellite systems [44].	Combined (time-shared) shadowed/unshadowed [44]

Table 2.2: Special Cases of Generalized Equation Corresponding to Specific Modulation/Demodulation Schemes

Detection Type	*Modulation (Signal Set)*	*Parameters of Marcum Q-Function*
Noncoherent	Equal energy, equiprobable correlated binary signals (λ=complex correlation coefficient)	$\eta = 1, a = \sqrt{\frac{1-\sqrt{1-\lvert\lambda\rvert^2}}{2}}$, $b = \sqrt{\frac{1+\sqrt{1-\lvert\lambda\rvert^2}}{2}}$
Noncoherent	Equal energy, equiprobable uncorrelated binary signals, e.g., BFSK	$\eta = 1, a = 0, b = 1$
Differentially Coherent	Binary phase-shift-keying (DPSK)	$\eta = 1, a = 0, b = \sqrt{2}$
Differentially Coherent	Quadrature phase-shift-keying (DQPSK) with Gray coding	$\eta = 1, a = \sqrt{2-\sqrt{2}}$, $b = \sqrt{2+\sqrt{2}}$

Chapter 3

LOSS DETECTION IN TCP CONGESTION CCONTROL

3.1 Current TCP

In the current TCP, two mechanisms have been deployed for the recovery of lost packets: the timeout mechanism and the Fast Retransmit and Recovery algorithm. A TCP sender uses packet loss as an implicit signal for network congestion with the assumption that packet losses are mainly caused by congestion. Packet losses are indicated by a timeout or by the receipt of three duplicated ACKs. The TCP sender continuously increases its traffic into the network until either one of above two indications is received. As shown in Figure 3.1, a retransmit timer timeout always forces TCP to enter the Slow-Start phase, during which the transmission rate is

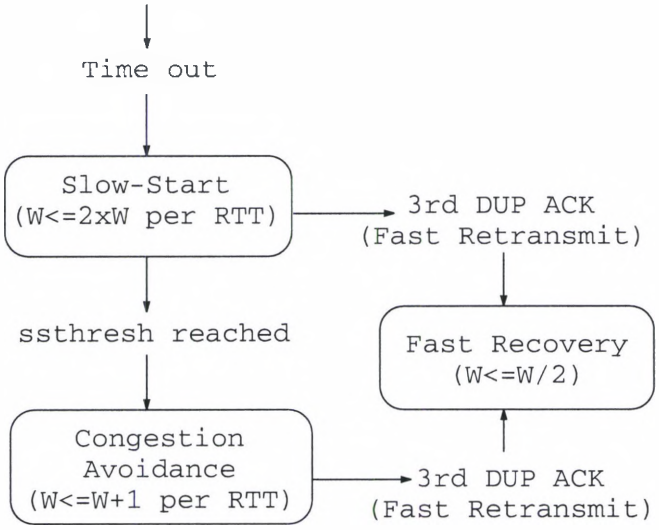

Figure 3.1: Current TCP states transition diagram.

doubled every *RTT* (Round Trip Time). When *ssthresh* is reached, TCP switches to Congestion Avoidance phase, during which the transmission rate increases linearly at one *MSS* (Maximum Segment Size) per *RTT*. The TCP sender will retransmit a packet if it receives three duplicated ACKs for the packet sent immediately before the lost packet. This procedure is called Fast Retransmit (See Figure 3.1). If the TCP sender detects a packet loss using Fast Retransmit, Fast Recovery will be used, which halves the congestion window. Congestion window size is denoted by W in Figure 3.1.

As mentioned previously, loss events in a network with lossy links are predominantly caused by link errors instead of congestion. Unfortunately, using current TCP, packet loss will be implicitly notified by either a timeout or the TCP source's receipt of three duplicated ACKs, which will always trigger the congestion control mechanism

of TCP. Since loss events are not always caused by network congestion, this operation will cause poor performance of TCP over lossy links.

3.2 ECN Marks

Active queue management mechanism, such as Random Early Detection (RED) [60, 61], allows a router to detect congestion before the queue overflows. Explicit Congestion Notification mechanism was proposed recently [20] to provide a fast and explicit indication of network congestion. Routers are therefore no longer limited to using packet drops to indicate congestion. This provides the basis for our proposed Diff-C-TCP, where we no longer use packet losses as the indicator of network congestion (more details could be found in Section 5.2). For ECN-capable TCP, if the sender receives an ECN_ECHO packet, then the sender knows that congestion was encountered on the path from the sender to the receiver. The receipt of ECN_ECHO packet indicating network congestion should be treated the same as a congestion loss in non-ECN-capable TCP. In other words, the congestion window should be halved and the slow start threshold *ssthresh* should be reduced. A router can indicate congestion by sending CE (Congestion Experienced) packets to the receiver. On receiving the CE packet, the TCP receiver will keep sending ECN_ECHO packet back to the sender till it receives CWR (Congestion Window Reduced) packet from TCP sender, which means the sender has already responded to network congestion. The sender does not react to the receipt of ECN_ECHO packets more than once every RTT.

Chapter 4

ELIMINATING CONGESTION LOSSES WITH ECN

If the buffer has enough space and the threshold of the RED router is appropriately set, zero-loss congestion control could be achieved by appropriately adjusting the source's window size based on the notification of ECN signal. In this section, we model the ECN mechanism and deduce the zero congestion loss requirements. As one of the most important parameters in our model, we also derive the exact mathematical expression for the average share of bottleneck link bandwidth by modeling the ECN marking dynamics as a Poisson Process. We finally end up with a complete mathematical model for achieving the zero-loss TCP congestion control.

4.1 Analysis Assumptions

In order to set up a proper but not complicated model and analyze the queue dynamics, as well as zero-loss requirements, we make the following assumptions:

- Senders always have data to send and will send as many as their windows allow.
- Receiver windows are large enough.
- Only one bottleneck link in the network causing queue build-up.
- There are no delayed acknowledgements.
- All packets have the same length.

4.2 Queue Dynamics Analysis with One TCP Flow

Along the path with only one TCP connection, if the bottleneck link bandwidth is d packet/second, then the downstream packet inter-arrival time and the acknowledgement inter-arrival time on the reserve link must be greater than or equal to $\frac{1}{d}$ [22].

Based on this statement, we use the analytical model shown in Figure 4.1 and notations as follows:

$w(t)$ = The sender's window size at time t.

r = Round Trip Time (RTT).

μ = Bandwidth of the bottleneck link.

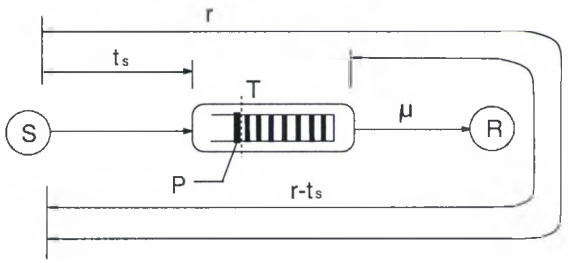

Figure 4.1: Analytical model of one TCP connection.

t_s = The time a packet needs to traverse from the sender to the router.

T = The threshold of RED router.

P = The packet which increases the queue length over T.

From the reported work in [22], we know that if at time t the bottleneck link has been busy for at least r seconds, and a packet just arrive at the congested router, the queue length at the congested router is

$$Q(t) = w(t - t_s) - r\mu. \tag{4.1}$$

Based on this theorem, requirements for achieving zero congestion loss could be described by the relationship between the queue length and the threshold of RED router. In two different phases, slow start and congestion avoidance, we have different results [22].

- Slow Start Phase

  The maximum queue length in slow start phase is $2T + r\mu + 1$.

- Congestion Avoidance Phase

The maximum queue length in congestion avoidance phase is $T+1$; the minimum queue length is $\frac{T-r\mu+1}{2}$. For the full utilization of the bottleneck link,

$$\frac{T - r\mu + 1}{2} \geq 0, \tag{4.2}$$

which means

$$T \geq r\mu - 1. \tag{4.3}$$

However, if $T > r\mu - 1$ indicating the minimum queue length is a positive number, the RED router may have long queueing delay. Thus, the optimal value of threshold for achieving zero congestion loss, full link utilization and minimum queueing delay in congestion avoidance phase can be calculated from

$$\frac{T - r\mu + 1}{2} = 0. \tag{4.4}$$

In other words, the optimal value of threshold is

$$T = r\mu - 1. \tag{4.5}$$

Finally, we end up with various cases of possible threshold shown in Table 4.1.

Table 4.1: The value of threshold T for the case of one TCP flow.

T	*Description*
$T > r\mu - 1$	The link is fully utilized, but packets suffer larger queueing delay.
$T = r\mu - 1$	Optimal threshold; zero congestion loss, full link utilization and minimum queueing delay.
$T < r\mu - 1$	The link is under-utilized.

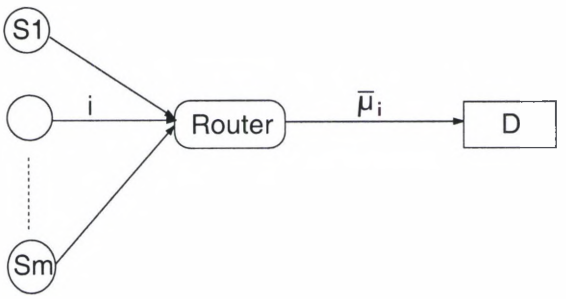

Figure 4.2: Analytical model of multiple competing TCP connections.

4.3 Queue Dynamics Analysis with Multiple Competing TCP Flows

Figure 4.2 shows the system model considered in this section. We use the following notations in the analysis:

$w_i(t)$ = The i^{th} sender's window size at time t.

r_i = Round Trip Time (RTT).

$\overline{\mu}_i$ = The average share of the bottleneck link bandwidth.

t_{s_i} = The time a packet needs to traverse from the i^{th} sender to the router.

T = The threshold of RED router.

The analysis of the case of multiple competing TCP connections is based on the discuss in Section 4.2. For the analytical model shown in Figure 4.2, we get the queue length expression as shown below.

$$Q(t) = \sum_{i=1}^{m} (w_i(t - t_{s_i}) - r_i \overline{\mu}_i). \tag{4.6}$$

As we mentioned in Chapter 1, the difficulty to use Equation 4.6 is how to calculate $\overline{\mu}_i$. We show the calculation of $\overline{\mu}_i$ in Section 4.4.

Similar to the discussion in Section 4.2, we give out the requirements for achieving zero congestion loss by two different cases, slow start phase and congestion avoidance phase [22].

- Slow Start Phase

  It is almost unlikely for multiple TCP connections to send at the same time. If some connections are in slow start phase, while others are in congestion avoidance phase, the queue length cannot be increased as fast as all connections are in slow start phase. If all connections start simultaneously, this could be considered as one aggregate flow. Accordingly, the maximum queue length is still $2T + r_i\overline{\mu}_i + 1$.

- Congestion Avoidance Phase

  The maximum queue length in congestion avoidance phase is $T + \alpha, \alpha \in [1, m]$; the minimum queue length is $\frac{T - r_i\overline{\mu}_i + \alpha}{2}, \alpha \in [1, m]$. Similarly, all possible thresh-

Table 4.2: The value of threshold T for the case of multiple TCP flows.

T	*Description*
$T > r\mu - 1$	The link is fully utilized, but packets suffer larger queueing delay.
$\mu - m \leq T \leq r\mu - 1$	Optimal threshold; zero congestion loss, full link utiliztion and minimum queueing delay.
$T < r\mu - m$	The link is under-utilized.

old values for the case of multiple TCP flows are shown in Table 4.2.

4.4 Modeling ECN Marks with Multiple TCP Flows

In this section, we show the calculation of the average share of the bottleneck link bandwidth which is the precondition of the zero-loss model we setup in the previous two sections. We consider the system model shown in Figure 4.2 and the Reno version of TCP. To keep the consistence, we use all the assumptions we made in the previous model.

Figure 4.3 shows the window evolution approximation for TCP-Reno sessions with different round trip delays sharing a bottleneck link with a RED gateway. This approximation has been used by many researchers for the analytic understanding of the RED performance, such as, authors in [23]. Based on Figure 4.3, we use the following notations in our model.

$w_{i,j}(t)$ = The j^{th} TCP session's window size right before the previous loss event.

$w_{i+1,j}(t)$ = The j^{th} TCP session's window size right before the current loss event.

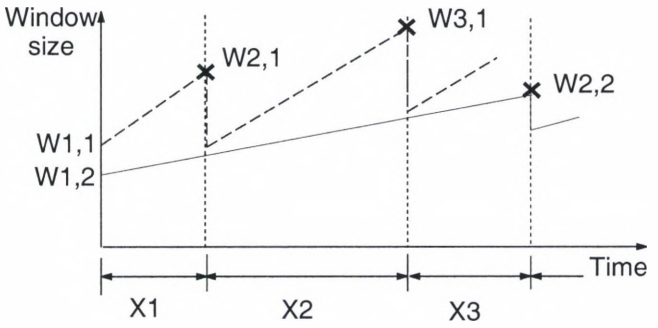

Figure 4.3: The window evolution approximation with two TCP-Reno flows.

$\overline{w_{j_{avg}}(t)}$ = The time-average window size for the j^{th} TCP session.

$r_{i,j}$ = Round Trip Time (RTT) of the j^{th} TCP session.

$\overline{\mu}_i$ = The average share of the bottleneck link bandwidth.

L_i = The time when i^{th} congestion loss event happens.

Consider the scenario in Figure 4.3; $X_i = L_i - L_{i-1}$ denotes the inter-loss duration. The window evolution could be approximately described by the following equation.

$$w_{i+1,j}(t) = \frac{w_{i,j}(t)}{2} + \frac{X_i}{r_{i,j}}. \tag{4.7}$$

Since loss events are determined by both the traffic type and the random marking of RED router, it is reasonable to consider $\{X_i\}$ as an Independent Identical Distributed (i.i.d.) *renewal process*. If we assume in any length of time interval, the number of loss event is Poisson distributed, then the total number of loss events in the interval $(0, t)$ is a *Poisson process*, denoted by $N(t)$. Therefore, the loss time interval X_i is an i.i.d. *exponential random variable* with the parameter λ. Its probability density

function (pdf) is

$$f_{X_i}(t) = \lambda e^{-\lambda t} u(t). \tag{4.8}$$

In addition, the waiting time $T[n] = \Sigma_{k=1}^{n} X_k$ for a loss event is a *gamma distributed random variable* with parameters (n, λ). Its pdf can be found as

$$f_T(t) = \frac{\lambda e^{-\lambda t} (\lambda t)^{n-1}}{\Gamma(n)} u(t), \tag{4.9}$$

which is, in the other form,

$$f_T(t) = \frac{\lambda^n e^{-\lambda t} t^{n-1}}{(n-1)!} u(t). \tag{4.10}$$

Based on the mathematical nature of the window evolution we analyzed above, we finally calculate the average share of bottleneck link bandwidth, which can be expressed as

$$\overline{\mu_i} = \frac{\overline{w_{j_{avg}}(t)}}{r_{i,j}}. \tag{4.11}$$

Taking the expectation for both sides of Equation 4.7, we have

$$\overline{w_{i+1,j}(t)} = \frac{\overline{w_{i,j}(t)}}{2} + \frac{\overline{X_i}}{r_{i,j}}. \tag{4.12}$$

Since any two loss events have the same statistical characteristics, it is apparent that

$w_{i+1,j}(t)$ and $w_{i,j}(t)$ have the same expected value. Thus,

$$\overline{w_j(t)} = 2\frac{\overline{X_i}}{r_{i,j}}. \tag{4.13}$$

Recall that the loss time duration is a renewal process and the total number of loss events during any length of time interval is a Poisson process, from Equation 4.8, we should have

$$\overline{X_i} = E[X_i] = \frac{1}{\lambda}. \tag{4.14}$$

Therefore,

$$\overline{w_j(t)} = \frac{2}{\lambda r_{i,j}}. \tag{4.15}$$

Because Poisson process is ergodic in mean (See Appendix for the proof), using the property of ergodicity, we have

$$\overline{w_{j_{avg}}(t)} = \overline{w_j(t)} = \frac{2}{\lambda r_{i,j}}. \tag{4.16}$$

Finally, the average share of the bottleneck link bandwidth is

$$\overline{\mu_i} = \frac{\overline{w_{j_{avg}}(t)}}{r_{i,j}} = \frac{2}{\lambda r_{i,j}^2}. \tag{4.17}$$

Remarkably, the above result implies that the connection with the shortest RTT has the largest average share, which is also found by authors in [22, 62].

Chapter 5

USING ECN TO IMPROVE THE TCP PERFORMANCE OVER LOSSY LINKS — DIFF-C-TCP

From the previous discussion, we can eliminate all network congestion losses (or the congestion losses are a small fraction of random losses) in the heterogeneous network environment involving lossy links. Therefore, with the negligible error all losses can be attributed to random losses. This leads to our proposed Diff-C-TCP discussed in this section.

5.1 Design Assumptions

Before we start to illustrate the principle of our proposed Diff-C-TCP algorithm, we make the following assumptions:

- Our proposed algorithm is used within a WAN or an enterprise network, where it is possible to make all routers and end-systems ECN-capable.
- When we mention wireless links, we do not look at mobility issues such as handoff or power requirements.

5.2 The Proposed Diff-C-TCP

In this section, we describe our proposed modification to TCP, called Differentiation Capable TCP (Diff-C-TCP).

Authors in [63] have pointed out that packet losses due to queue buffer overflows is relatively infrequent when a majority of end-systems become ECN-capable and participate in TCP or other compatible congestion control mechanisms. Furthermore, we are able to use the model we set up in the previous sections to eliminate congestion losses. In addition, link errors become more significant compared to packet losses due to buffer overflows in wireless links. Therefore, it is reasonable to assume that all loss events come from links errors, **unless** some congestion indications could be found.

Our proposed algorithm assumes that packet losses indicate corruption, and the TCP sender uses ECN as an explicit signal of network congestion. Diff-C-TCP's

response to ECN is similar to TCP's response to packet losses. In other words, the receipt of ECN packets should trigger a response to network congestion. Packet losses are treated as link errors unless ECN packets are received.

Figure 5.1 shows the kernel of our proposed Diff-C-TCP algorithm at the sender's side. A Diff-C-TCP sender treats the situation that the retransmit timer times out without receiving any ECN_ECHO packet and (or) receiving duplicate acknowledgements as the indication of link errors. Most often, this is the case in a network with wireless links (packet losses due to link errors). In this case, the Diff-C-TCP source does not decrease *cwnd*. If the Diff-C-TCP sender receives the ECN_ECHO packet sent by the receiver, the sender treats it as network congestion and triggers the Fast Recovery algorithm [20] as in the current TCP.

In Diff-C-TCP, the congestion window size is appropriately controlled in the presence of either network congestion or corruption. Congestion window is halved using Fast Recovery algorithm when there is network congestion (explicitly notified by ECN_ECHO packets), and persists at the previous value in the presence of corruption. There are two mechanisms that might be applied to adjust congestion window when Diff-C-TCP sender detects corruption: (i) keep *cwnd* unchanged as the previous value; (ii) use Congestion Avoidance algorithm to slowly increase *cwnd*. In our algorithm, we adapt the first mechanism —- make congestion window persist in the previous value.

ECN mechanism will be most effective if it is used with active queue management

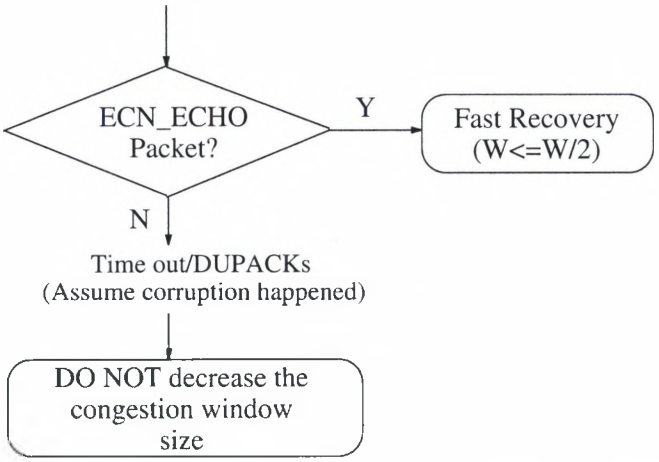

Figure 5.1: States transition diagram of Diff-C-TCP for Diff-C-TCP kernel.

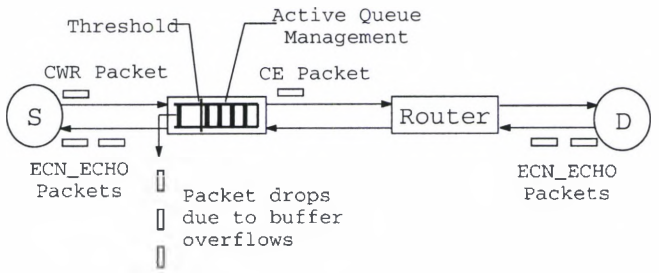

Figure 5.2: A simple Diff-C-TCP based model.

(such as RED) [20] as illustrated in Figure 5.2. In active queue management, when a buffer reaches a certain threshold, the router will send a CE packet to the TCP receiver. Routers send CE packets before their buffers overflow. Therefore, packet drops due to congestion happen only after the router has sent CE packets. Upon receiving the CE packet, the TCP receiver will keep sending ECN_ECHO packet back to the sender until it receives a CWR packet from the sender, which means the sender has responded to network congestion. The sender only responds to the first ECN_ECHO packet and ignores others up to one RTT. Depending on the threshold

of RED and the level of network congestion, ECN_ECHO packets can arrive at the sender either before or after the retransmit timer times out due to congestion packet losses (caused by buffer overflow). Our proposed Diff-C-TCP is effective in both the above cases as described below.

5.2.1 Case 1: Retransmit Timer Times Out After ECN_ECHO Packets Are Received By the Sender.

In this case, the sender has already responded to congestion which had been indicated by the receipt of ECN_ECHO packets. This case is desirable.

5.2.2 Case 2: Retransmit Timer Times Out Before ECN_ECHO Packets Are Received By the Sender.

In this case, as shown in Figure 5.3, the retransmit timer timeout due to buffer overflow happens at time t_1, and ECN_ECHO packets are received by the sender at time t_2. If the difference between t_1 and t_2 is small enough, though the TCP sender does not respond to packet losses indicated by retransmit timer timeout at t_1 (according to our Diff-C-TCP), ECN_ECHO packets will arrive very quickly, which will trigger Fast Recovery mechanism to get the network out of congestion.

The difference between t_1 and t_2 can be decreased by decreasing t_2 as described below. As mentioned above, using active queue management such as RED, when

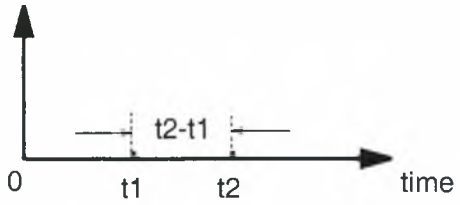

Figure 5.3: Time sequence of Case 2.

buffer reaches threshold, it will send the CE packet to receiver. Upon receiving the CE packet, the receiver starts to send ECN_ECHO packets to the sender. Though it is difficult to control the travel time of ECN_ECHO packets from the receiver to the sender, we can make the receiver send ECN_ECHO packets earlier by letting the router send the CE packet earlier. The earlier ECN_ECHO packets are sent, the earlier they arrive at the sender, i.e., the smaller the value of t_2 is. The time when the router sends the CE packet is decided by the value of threshold. Therefore, an optimum value of RED's threshold is very important and expected, which is one of our current research topics.

Chapter 6

PERFORMANCE EVALUATION

6.1 Simulation Methodology

We have evaluated the performance of our Diff-C-TCP algorithm using the *ns* (*ns* Version 2.1b6) simulation tool from Berkeley [64]. The implementation is based on RFC 2481 [20], the current TCP with ECN capability and our proposed Diff-C-TCP. Our network topology for conducting simulations is shown in Figure 6.1. Two local area network (10 Mbps) are connected using a satellite link (64Kbps) with a propagation delay of 280ms. Like previous researchers, a Uniform random error model is used to generate random errors on the satellite link [65, 66]. Instead of

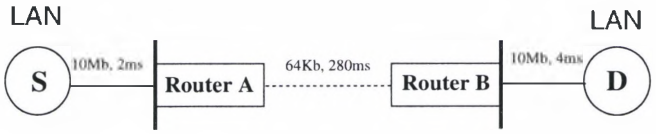

Figure 6.1: LAN interconnection using a satellite link.

dropping packets at routers, Random Early Detection (RED) routers are used in our simulations to set the CE (Congestion Experienced) bit.

All the links in Figure 6.1 are labeled with a (bandwidth, propagation delay) pair. The full-duplex link between router A and router B has a bit-error rate BER, which varies between $1e^{-7}$ to $1.2e^{-4}$ in our simulation. The receiver's advertised window size (initial *ssthresh*) is set to 30 segments. The packet size is set to 1000 bytes (when BER is below $5e^{-5}$) or 512 bytes (when BER exceeds $5e^{-5}$).

To ensure a fair comparison between the performance of the current TCP with ECN capability and Diff-C-TCP, we use the above in our simulation. The reason is as follows. For a certain packet size, an increase in the BER results in a high number of packets being dropped due to link errors. If the BER is increased sufficiently enough, every packet passing through the link will probably have errors and will be dropped, thereby hurting the measurement. However, if we decrease the packet size when the BER reaches a certain high value, fewer packets will be dropped for the same value of BER, which makes it possible to consider the effectiveness of our algorithm under high BER scenario.

Ftp was used in our simulation to transfer data from the source to destination. We used a method called dynamic test to make the throughput measurement easier to control. Instead of fixing the size of the file to be transferred, we controlled the simulation time as a function of the BER. By changing three parameters, viz, the number of packets being dropped, simulation time and packet size, we could carry

out simulations for a wide range of BER values. This resulted in a flexible and efficient way to avoid a lot of difficulties in realizing simulations under high BER conditions. In the next section, we present results regarding the effectiveness of our algorithm.

6.2 Simulation Results

In this section, results obtained from simulation experiments for both Diff-C-TCP and the current TCP with ECN capability are shown. We compare the performance of Diff-C-TCP and the current TCP with ECN capability by analyzing their congestion window size and throughput.

6.2.1 Congestion Window Size: cwnd

We used the network configuration and simulation methodology described in Section 6.1 to perform this simulation. Figure 6.2 shows the number of data packets transmitted using the current TCP with ECN capability. There are two separate marks for each packet: one when packets arrive at the router and one when they leave the router. The x axis represents time, and the y axis shows the packet number mod 90. Each dropped packet is denoted by $\times$. The ECN_ECHO packet is denoted by a small open square. Each CWR packet is denoted by a solid diamond. It is seen that multiple packet losses due to corruption happen at time 57 sec, and network congestion happens at 19 sec, 22 sec, 34 sec and 50 sec. Figure 6.3 shows the congestion window

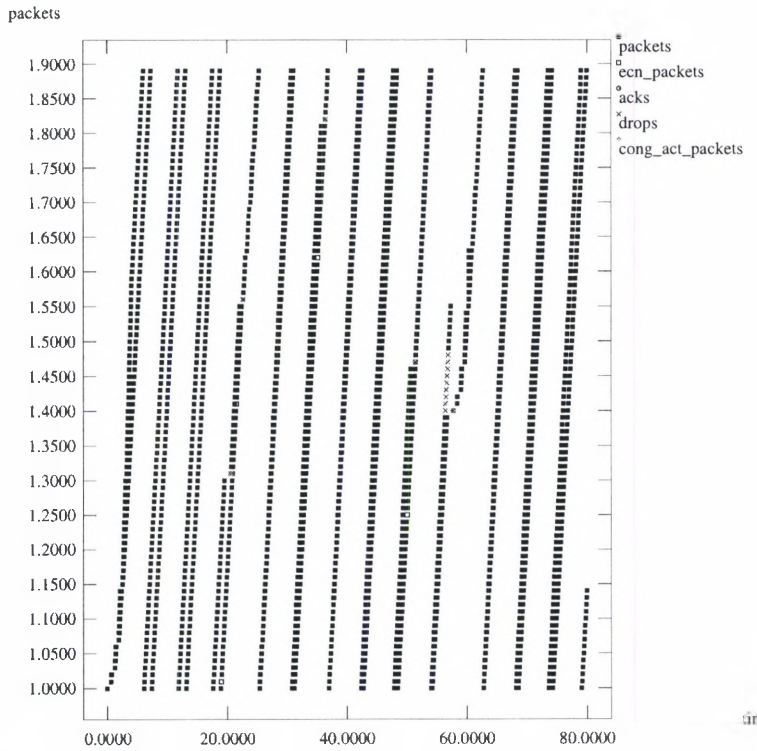

Figure 6.2: Number of packets transmitted for the current TCP with ECN capability.

at the sender. As seen in Figure 6.3, at time of 57 sec, the congestion window is reduced to the initial value due to triggering of Slow-Start mechanism (in the current TCP with ECN capability) arising from corruption losses. It is also seen that network congestion at 19 sec, 22 sec, 34 sec and 50 sec result in the congestion window being halved due to triggering of Fast Recovery mechanism. Figure 6.5 shows the congestion window when Diff-C-TCP is applied with the packet transmission shown in Figure 6.4. It is seen that the congestion window doesn't go down when multiple packet losses happen at 57 sec, but is halved at 19 sec, 22 sec, 34 sec, 50 sec and 68 sec.

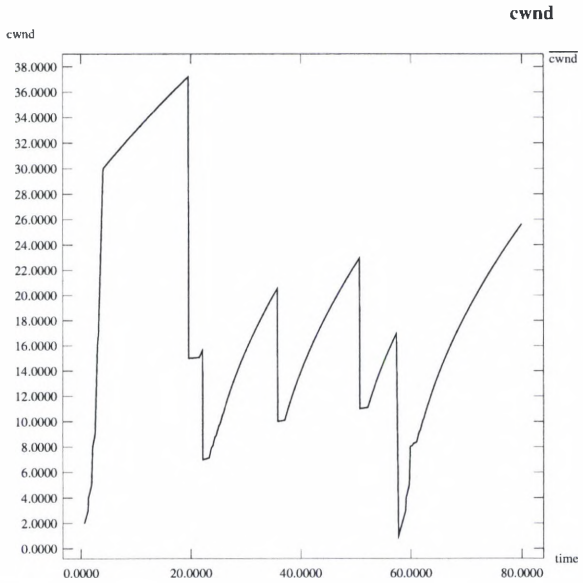

Figure 6.3: Congestion window size for the current TCP with ECN capability.

Comparing Figures 6.5 and 6.3, we find that, when Diff-C-TCP is used, the congestion window never goes down in the presence of packet losses due to link errors; instead, it persists in the previous value as mentioned in the Diff-C-TCP algorithm (See Section 5.2). The Fast Recovery mechanism is triggered in the presence of network congestion (notified by ECN_ECHO packets). However, when the current TCP (with ECN capability) is used in the simulation, the timeout caused by packet losses triggers the Slow-Start mechanism that results in the reduction of congestion window to the initial value. The reason is that the current TCP (though it is ECN capable in our simulation) makes the assumption that all losses are caused by congestion. Thus, when there is the packet loss, no matter whether it is caused by congestion or corruption, the current TCP with ECN capability triggers TCP's congestion control mechanism. Since Diff-C-TCP assumes by default that losses are due to link errors,

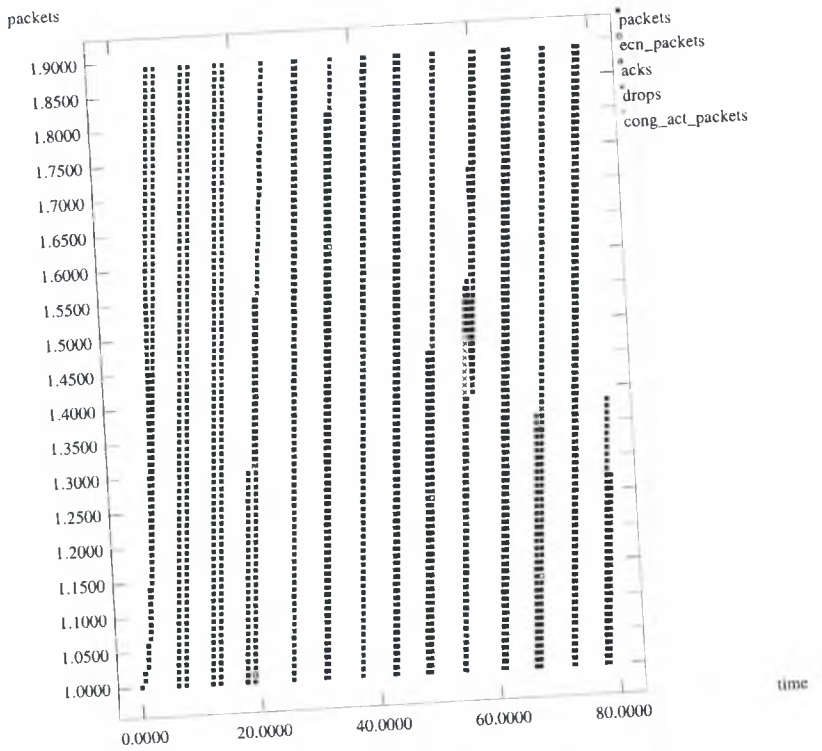

Figure 6.4: Number of packets transmitted for Diff-C-TCP.

and network congestion is explicitly notified by ECN_ECHO packets, TCP's congestion control mechanism will not be triggered when packets are lost due to link errors. It can only be triggered by the receipt of ECN_ECHO packets, as shown in Figure 6.5 (combined with Figure 6.4).

6.2.2 Throughput and Goodput

In this test, we compare the throughput of Diff-C-TCP and the current TCP with ECN capability. We used the same network parameters as in the previous test (See Section 6.1) and used the dynamic test method to measure the *goodput* (bit/s) (the

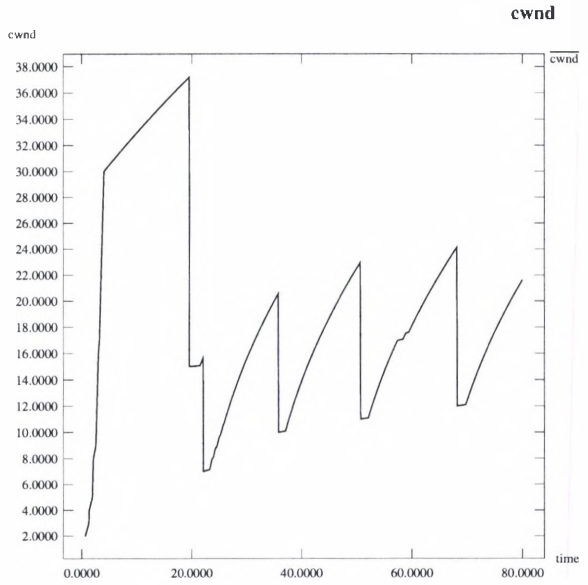

Figure 6.5: Congestion window size for Diff-C-TCP.

amount of useful information, in *bit*, being received by the receiver per second, not including errors) and the *normalized throughput* of both the TCPs for BER values ranging from $1e^{-7}$ to $1.2e^{-4}$.

Figure 6.6 compares the goodput in bit/s of both Diff-C-TCP and the current TCP with ECN capability. The normalized throughput of both the TCPs are shown in Figure 6.7. We see that Diff-C-TCP's throughput is much higher than that of the current TCP with ECN capability. At a BER of $5e^{-5}$, the goodput of our Diff-C-TCP is almost **5 times** higher than that of the current TCP. From Figures 6.6 6.7, this improvement is much higher at higher values of BER. In addition, the throughput of the current TCP with ECN suffers more severely than our Diff-C-TCP as the error rate increases. We can also see that, with the increase of the value of BER, the throughput of the current TCP with ECN capability decreases much faster than the

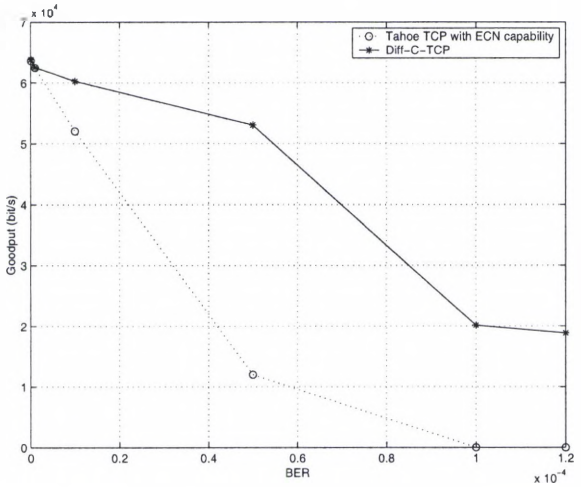

Figure 6.6: Comparison of goodput (bit/s).

throughput of our Diff-C-TCP. For example, according to Figure 6.6, when the value of BER increases from $1e^{-5}$ to $5e^{-5}$, the current TCP's goodput decreases by **77 %** in contrast to our Diff-C-TCP whose goodput only decreases by **12 %**. This is also valid for normalized throughput as shown in Figure 6.7.

As mentioned previously, the current TCP with ECN used in our simulation makes an assumption by default that every loss event is caused by network congestion and a congestion control algorithm is triggered. As a result, the congestion window size must be reduced. Therefore, each loss event, irrespective of whether it is due to congestion or corruption, will affect the throughput and lead to lower throughput compared to Diff-C-TCP's. In the case of Diff-C-TCP, all packet losses are assumed to be caused by link errors and network congestion is indicated by the receipt of ECN_ECHO packets. The congestion window will not be changed in the presence of packet losses due to corruption. Thus, only network congestion can affect its throughput. Furthermore,

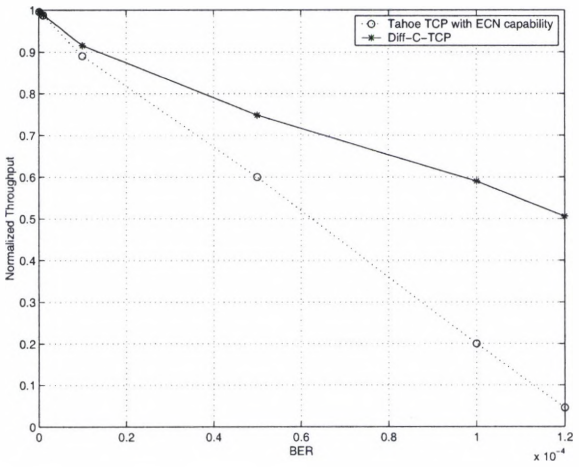

Figure 6.7: Comparison of normalized throughput.

for the current TCP with ECN, a higher value of BER results in more packets dropped and more frequent reduction of the congestion window. Because of this, at higher values of BER, the frequency of reduction of the congestion window of the current TCP with ECN is so high that it is almost impossible for the congestion window size to reach a high value. This eventually hurts the throughput, because the value of congestion window size is not very high even during the *no loss* period. It is totally different when our Diff-C-TCP is applied, because the congestion window size is only affected by real congestion that is explicitly notified by ECN_ECHO packets. As described in Section 5.2 and shown in Figure 6.5, packet losses due to corruption never change the congestion window size.

Chapter 7

CONCLUSIONS AND FUTURE WORK

7.1 Conclusions

In this thesis, we first presented a novel approach to classify the existing wireless error modeling methods, based on different requirements of different researchers in the practical engineering. A number of error models used to approximate the loss behavior of transmission channels have been classified into two groups: *analytical models* and *empirical distribution-based models.* According to our classification, *we recommend* that researchers who need to analyze characteristics of wireless error channels should use *Analytical Models*; network protocol designers and evaluators who need to synthesize a wireless error channel, i.e., build an appropriate error model for

their simulations and evaluations over special communication environments, should use *Empirical Distribution-Based Models.*

We have set up a complete model to achieve zero congestion losses with multiple competing TCP flows in the heterogeneous network involving lossy links. In addition, we have also deduced the exact mathematical expression of the average share of the bottleneck link bandwidth, one of the most important parameters in our model, which has been found by other researchers to be the biggest difficulty.

Finally, having both the extensive understanding of wireless links and the possibility of zero-loss TCP congestion control, we proposed a new TCP algorithm (named Diff-C-TCP) to improve the TCP throughput performance in the presence of non-congestion related losses. Diff-C-TCP differentiates between losses due to network congestion and corruption on lossy links by assuming that all loss events are caused by link errors, unless the network explicitly informs the sources of congestion. Network congestion is explicitly indicated by the receipt of ECN_ECHO packets. Whenever the TCP retransmit timer times out without the sender receiving any ECN_ECHO packet, the congestion window in Diff-C-TCP persists in its previous value as a response to corruption. When the Diff-C-TCP sender receives an ECN_ECHO packet, Fast Recovery algorithm is triggered as a response to network congestion.

The proposed Diff-C-TCP has been studied in detail using simulation and has been found to significantly improve TCP performance on the lossy satellite link. Our simulation results have shown that Diff-C-TCP does not reduce the congestion

window in the presence of packet loss due to corruption. Congestion window is changed only due to network congestion. We have also achieved significant goodput improvement, up to **5 times**, over the current TCP with ECN for data transfer across a typical satellite link with high bit-error rates.

7.2 Recommendations for Future Work

We have developed a new TCP, named Diff-C-TCP, to improve the TCP performance over wireless links. As we have shown, Diff-C-TCP produces much higher throughput than the current TCP does over the lossy environment. However, we find that our current version of Diff-C-TCP is too aggressive to be used in the global Internet. In this section, we describe this issue, and accordingly, make some recommendations for the future modifications to our current scheme.

7.2.1 ECN Compatibility

The development of our Diff-C-TCP algorithm is based on the assumption that it is used within a WAN or an enterprise network, where it is possible to make all routers and end-systems ECN-capable. However, if it is used in the global Internet, routers that do not generate ECN_ECHO packets would be highly incompatible with Diff-C-TCP, because zero-loss congestion control is proposed in this thesis to be implemented with ECN mechanism. In such a case, packet droppings due to buffer overflows must

happen. If a Diff-C-TCP session was to operate over such a router, then the session would not respond to the router dropping packets due to congestion. Instead, the Diff-C-TCP session would interpret the packet loss due to a link error, and continue to transmit at a constant rate. This would cause dead-lock at the router.

Therefore, to find a solution to the ECN compatibility problem has the first preference among future works.

7.2.2 The Consideration of Mobility

Another assumption made for the development of Diff-C-TCP is that only the loss behavior of wireless links is considered. In the future work, the hand-off problem of wireless networks should be considered. During the hand-offs and situations where long fades occur, such as turning a corner and entering a region of heavy shadowing by some obstruction, there could be long bursts of packet loss, or long periods of reduced bandwidth on the wireless channel, in which case the effective bandwidth of the wireless link has been reduced, and therefore the offered load (the sender's congestion window size) must also be reduced.

In such a case, an algorithm which is able to differentiate packet losses due to hand-offs and packet losses due to congestion is to be developed. This might be an investigation of delay characteristics of TCP sessions.

7.2.3 Quantitative Benefits of ECN for TCP over Wireless

People used to use computer simulation to test the performance of existing or proposed algorithms. But what are the quantitative benefits of ECN for TCP over wireless? What new dynamics does ECN add to TCP, in terms of competition between ECN-capable and non-ECN-capable traffic, the robustness of ECN in the presence of dropped ACK packets, performance with multiple congested gateways, etc.? Apparently, this needs much modeling work on both ECN performance and TCP over wireless.

Bibliography

[1] S. Dawkins, G. Montenegro, M. Kojo, V. Magret, and N. Vaidya, "End-to-end performance implications of links with errors." draft-ietf-pilc-error-03.txt, March 2000.

[2] N. K. G. Samaraweera and G. Fairhurst, "Reinforcement of TCP error recovery for wireless communication," *Computer Communication Review*, vol. 28, no. 2, April 1998.

[3] H. Balakrishnan, S. Seshan, E. Amir, and R. H. Katz, "A comparison for improving TCP/IP performance over wireless linkss," *ACM SIGCOMM*, Stanford, CA, August 1996.

[4] V. Jacobson, "Congestion avoidance and control," *ACM SIGCOMM*, 1988.

[5] R. Braden, "Requirements for internet hosts communication layers." RFC 1122, October 1989.

[6] W. Richard Stevens, "TCP slow start, congestion avoidance, fast retransmit, and fast recovery algorithm." RFC 2001, January 1997.

[7] M. Allman, "Improving TCP performance over satellite channels," master's thesis, Ohio University, June 1997.

[8] M. Allman, C. Hayes, H. Kruse, and S. Ostermann, "TCP performance over satellite links," *5th Int'l Conference on Telecommunication Systems*, Nashville, March 1997.

[9] J. C. Hoe, "Improving the start-up behavior of a congestion control scheme for TCP," *SIGCOMM*, California, USA, pp. 270–280, ACM, 1996.

[10] S. Floyd, "TCP and successive fast retransmits," technical report, Lawrence Berkeley Laboratory, October 1994.

[11] R. Goyal, R. Jain, S. Kalyanaraman, S. Fahmy, and S. Ch. Kim, "Improving the performance of TCP over the ATM-UBR service," *ICC*, Montreal, pp. 1042–1048, June 1997.

[12] K. Fall and S. Floyd, "Simulation-based comparison of Tahoe, Reno, and SACK TCP," *Computer Communication Review*, vol. 26, no. 3, pp. 5–21, July 1996.

[13] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow, "TCP selective acknowledgment options." RFC 2018, October 1996.

[14] M. Allman, S. Dawkins, D. Glover, J. Griner, T. Henderson, J. Heidemann, H. Kruse, S. Ostermann, K. Scott, J. Semke, J. Touch, and D. Tran, "Ongoing TCP research related to satellites." draft-ietf-tcpsat-res-issues-12.txt, October 1999.

[15] S. Biaz and N. Vaidya, "Using end-to-end statistics to distinguish congestion and corruption losses: A negative result," Technical report 97-009, Texas A&M University, August 1997.

[16] S. Biaz and N. Vaidya, "Sender-based heuristics for distinguishing congestion losses from wireless transmission losses," Technical report 98-013, Texas A&M University, June 1998.

[17] S. Biaz and N. Vaidya, "Discriminating congestion losses from wireless losses using inter-arrival times at the receiver," Technical report 98-014, Texas A&M University, June 1998.

[18] S. Banerjee, J. Ram Goteti, and G. Krishnamoorthy, "Extending TCP for wireless networks." http://www.cs.umd.edu/users/suman/docs/711s97/711s97.html, May 1997.

[19] S. Floyd, "TCP and explicit congestion notification," *ACM Computer Communication Review*, vol. 24, no. 5, pp. 10–23, October 1994.

[20] K. Ramakrishnan and S. Floyd, "A proposal to add Explicit Congestion Notification (ECN) to IP." RFC 2481, January 1999.

[21] S. Kunniyur and R. Srikant, "End-to-end congestion control schemes: Utility functions, random losses and ECN marks," *INFOCOM*, Tel Aviv, Israel, IEEE, March 2000.

[22] C. Liu and R. Jain, "Buffer requirements for zero-loss flow control with explicit congestion notification." http://www.cis.ohio-state.edu/ jain/papers/ecnbuffer.htm, September 1999.

[23] A. Abouzeid and S. Roy, "Analytic understanding of RED gateways with mutiple competing TCP flows," *GLOBECOM*, San Francisco, CA, IEEE, November 2000.

[24] J. B. Andersen, T. S. Rappaport, and S. Yoshida, "Propagation measurements and models for wirelsss communication channels," *IEEE Commun. Mag.*, vol. 33, no. 1, pp. 42–49, January 1995.

[25] T. S. Rappaport, *Wireless Communications*, Upper Saddle River, NJ: Prentice Hall, 1996.

[26] B. Sklar, "Rayleigh fading channels in mobile digital communication systems part i: Characterization," *IEEE Commun. Mag.*, vol. 35, no. 9, pp. 136–146, September 1997.

[27] F. Babich and G. Lombardi, "Statistical analysis and characterization of the indoor propagation channel," *IEEE Trans. Commun.*, vol. 48, no. 3, pp. 455–464, March 2000.

[28] W. C. Jakes, *Microwave Mobile Communications*, New York: Wiley, 1974.

[29] M. K. Simon and M. S. Alouini, "A unified approach to the performance analysis of digital communication over generalized fading channels," *Proc. IEEE*, vol. 86, no. 9, pp. 1860–1877, September 1998.

[30] L. Rayleigh, "On the resultant of a large number of vibrations of the same pitch and of arbitrary phase," *Phil. Mag.*, vol. 27, no. 6, pp. 460–469, June 1889.

[31] R. S. Hoyt, "Probability functions for the modulus and angle of the normal complex variate," *Bell Syst. Tech. J.*, vol. 26, no. 4, pp. 318–359, April 1967.

[32] S. O. Rice, "Statistical properties of a sine wave plus random noise," *Bell. Syst. Tech. J.*, vol. 27, no. 1, pp. 109–157, January 1948.

[33] M. Nakagami, *Statistical Methods in Radio Wave Propagation*, pp. 3–36. Oxford, England: Pergamon, 1960.

[34] H. Suzuki, "A statistical model for urban multipath propagation," *IEEE Trans. Commun.*, vol. COM-25, no. 7, pp. 673–680, July 1977.

[35] M. J. Ho and G. L. Stber, "Co-channel interference of microcellular systems on shadowed nakagami fading channels," *Proc. IEEE Vehicular Technology Conf. VTC'93*, Secaucus, NJ, May 1993.

[36] M. K. Simon and M. S. Alouini, "A unified approach to the probability of error for noncoherent and differentially coherent modulations over generalized fading channels," *IEEE Trans. Commun.*, vol. 46, no. 12, pp. 1625–1638, December 1998.

[37] H. B. James and P. I. Wells, "Some tropospheric scatter propagation measurements near the radio-horizon," *Proc. IRE*, pp. 1336–1340, October 1955.

[38] B. Chytil, "The distribution of amplitude scintillation and the conversion of scintillation indices," *J. Atmos. Terr. Phys.*, vol. 29, no. 9, pp. 1175–1177, September 1967.

[39] R. J. C. Bultitude, S. A. Mahmoud, and W. A. Sullivan, "A comparison of indoor radio propagation characteristics at 910 MHz and 1.75 GHz," *IEEE J. Select Areas Commun.*, vol. 7, no. 1, pp. 20–30, January 1989.

[40] T. Aulin, "Characteristics of a digital mobile radio channel," *IEEE Trans. Veh. Technol.*, vol. VT-30, no. 5, pp. 45–53, May 1981.

[41] G. L. Stber, *Principles of Mobile Communications*, ch. sec. 2.4. Norwell, MA: Kluwer, 1996.

[42] T. S. Pappaport, S. Y. Seidel, and K. Takamizawa, "Statistical channel impulse response models for factory and open plan building radio communication system design," *IEEE Trans. Coommun.*, vol. 39, no. 5, pp. 794–807, May 1991.

[43] M. J. Ho and G. L. Stber, "Co-channel interference of microcellular systems on shadowed nakagami fading channels," *Proc. IEEE Veh Technol. Conf. VTC'93*, Secaucus, NJ, pp. 568–571, May 1993.

[44] E. Lutz, D. Cygan, M. Dippold, F. Dolainsky, and W. Papke, "The land mobile satellite communication channel–recording, statistics and channel model," *IEEE Trans. Veh. Technol.*, vol. VT-40, no. 5, pp. 375–386, May 1992.

[45] J. G. Proakis, "On the probability of error for multichannel reception of binary signals," *IEEE Trans. Commun. Technol.*, vol. COM-16, no. 2, pp. 68–71, February 1968.

[46] W. C. Lindsey, "Error probability for ricean fading multichannel reception of binary and n-ary signals," *IEEE Trans. Inform. Theory*, vol. IT-10, no. 10, pp. 339–350, October 1964.

[47] U. Charash, "Reception through nakagami fading multipath channels with random delay," *IEEE Trans. Commun.*, vol. COM-27, no. 4, pp. 657–670, April 1979.

[48] J. F. Weng and S. H. Leung, "Analysis of DPSK with equal gain combining in nakagami fading channels," *Electron. Lett.*, vol. 33, no. 4, pp. 654–656, April 1997.

[49] F. Patenaude, J. H. Lodge, and J. Y. Chouinard, "Error probability expressions for noncoherent diversity in nakagami fading channels," *Proc. IEEE Vehicular Technology Conf. VTC'97*, Phoenix, AZ, pp. 1484–1487, May 1997.

[50] T. T. Tjhung, C. Loo, and N. P. Secord, "BER performance of DQPSK in slow rician fading," *Electron. Lett.*, vol. 28, no. 8, pp. 1763–1765, August 1992.

[51] M. Tanda, "Bit error rate of DQPSK signals in slow nakagami fading," *Electron. Lett.*, vol. 29, no. 3, pp. 431–432, March 1993.

[52] C. Tellambura and V. K. Bhargava, "Unified error analysis of DQPSK in fading channels," *Electron. Lett.*, vol. 30, no. 12, pp. 2110–2111, December 1994.

[53] W. Turin, *Digital Transmission System: Performance Analysis and Modeling*, New York: McGraw-Hill, 1998.

[54] E. N. Gillbert, "Capacity of a bursty-noise channel," *Bell Syst. Tech. J.*, vol. 39, no. 9, pp. 1253–1265, September 1960.

[55] E. O. Elliott, "Estimates of errors rates for codes on bursty-noise channels," *Bell Syst. Tech. J.*, vol. 42, no. 9, pp. 1977–1997, Spetember 1963.

[56] J. R. Yee and E. J. Weldon, "Evaluation of the performance of error correcting codes on a gilbert channel," *IEEE Trans. Coommun.*, vol. 43, no. 8, pp. 2316–2323, August 1995.

[57] M. Zorzi and R. R. Rao, "Lateness of probability of a retransmission scheme for error control on a two-state markov channel," *IEEE Trans. Commun.*, vol. 47, no. 10, pp. 1537–1548, October 1999.

[58] H. S. Wang and N. Moayeri, "Finite-state markov channel–a useful model for radio communication channels," *IEEE Trans. Veh. Technol*, vol. 44, no. 1, pp. 163–171, February 1995.

[59] G. T. Nguyen, B. Noble, R. H. Katz, and M. Satyanarayanan, "A trace-based approach for modeling wireless channel behavior," *Proc. Wint. Simu. Conf.*, December 1996.

[60] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, no. 4, pp. 397–413, August 1993.

[61] B. Braden, D. Clark, J. Crowcroft, B. Davie, S. Deering, D. Estrin, S. Floyd, V. Jacobson, G. Minshall, C. Partridge, L. Peterson, K. Ramakrishnan, S. Shenker, J. Wroclawski, and L. Zhang, "Recommendations on queue management and congestion avoidance in the internet." RFC 2309, April 1998.

[62] A. Mistra, T. Ott, and J. Baras, "The window distribution of multiple TCPs with random loss queues," *GLOBECOM*, Rio de Janeiro, Brazil, IEEE, December 1999.

[63] K. K. Ramakrishnan, S. Floyd, and D. Black, "The addition of Explicit Congestion Notification (ECN) to IP." draft-ietf-tsvwg-ecn-00.txt, November 2000.

[64] VINT Project U.C. Berkeley/LBNL, "ns v2.1b6: Network simulator." http://www-mash.cs.berkeley.edu/ns/, January 2000.

[65] H. Balakrishnan, V. Padmanabhan, S. Seshan, and R. Katz, "A comparison of mechanisms for improving TCP performance over wireless links," *IEEE/ACM Transactions on Networking*, vol. 6, no. 5, pp. 756–769, December 1997.

[66] C. Parsa and J.J. Garcia-Luna-Aceves, "TULIP: A link-level protocol for improving TCP over wireless links," *WCNC*, New Orleans, Louisana, pp. 21–24, IEEE, September 1999.

Appendix A

THE PROOF OF THE ERGODICITY OF POISSON PROCESS

Poisson process can be expressed as (See Figure A.1)

$$N(t) = \sum_{0}^{\infty} u(t - T[n]). \tag{A.1}$$

As mentioned in Section 4.4, $T[n] = \sum_1^n X_k$ is the arrival time for a loss event. Because $N(t)$ is a *Poisson process*, during any time internal length of t, say, $(0, t)$, the total number of events is Poisson distributed with *mean* λt.

Define a random sequence $Y[n]$ = the total number of events happened in the time interval $(0, t)$, according to the above analysis,

$$E[Y[n]] = \lambda t. \tag{A.2}$$

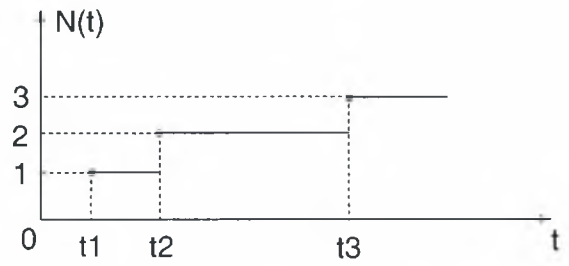

Figure A.1: Poisson process.

Then, according to *Strong Law of Large Numbers*,

$$\lim_{n\to\infty} \frac{1}{n} \sum_{k=1}^{n} Y[k] = \lambda t. \tag{A.3}$$

Thus,

$$\lim_{n\to\infty} \frac{1}{n} \sum_{k=1}^{n} Y[k] = \lambda t = E[N(t)]. \tag{A.4}$$

In other words, the time average is equal to the expected value. Therefore, Poisson process is ergodic in mean.

Appendix B

TCL/TK CODES OF PERFORMANCE TESTS SIMULATION

```
set dir [pwd]
catch "cd tcl/test"
source misc_simple.tcl
catch "cd $dir"

set flowfile fairflow.tr; # file where flow data is written
set flowgraphfile fairflow.xgr; # file given to graph tool

Class Topology

Topology instproc node? num {
    $self instvar node_
    return $node_($num)
}

Topology instproc makenodes ns {
    $self instvar node_
    set node_(s1) [$ns node]
    set node_(s2) [$ns node]
    set node_(r1) [$ns node]
    set node_(r2) [$ns node]
    set node_(s3) [$ns node]
    set node_(s4) [$ns node]
}
```

```
Topology instproc createlinks ns {
    $self instvar node_
    $ns duplex-link $node_(s1) $node_(r1) 10Mb 2ms DropTail
    $ns duplex-link $node_(s2) $node_(r1) 10Mb 3ms DropTail
    $ns duplex-link $node_(r1) $node_(r2) 64Kb 280ms RED
    $ns queue-limit $node_(r1) $node_(r2) 25
    $ns queue-limit $node_(r2) $node_(r1) 25
    $ns duplex-link $node_(s3) $node_(r2) 10Mb 4ms DropTail
    $ns duplex-link $node_(s4) $node_(r2) 10Mb 5ms DropTail

    $ns duplex-link-op $node_(s1) $node_(r1) orient right-down
    $ns duplex-link-op $node_(s2) $node_(r1) orient right-up
    $ns duplex-link-op $node_(r1) $node_(r2) orient right
    $ns duplex-link-op $node_(r1) $node_(r2) queuePos 0
    $ns duplex-link-op $node_(r2) $node_(r1) queuePos 0
    $ns duplex-link-op $node_(s3) $node_(r2) orient left-down
    $ns duplex-link-op $node_(s4) $node_(r2) orient left-up
}

Class Topology/net2 -superclass Topology
Topology/net2 instproc init ns {
    $self instvar node_
    $self makenodes $ns
    $self createlinks $ns
}

Class Topology/net2-lossy -superclass Topology
Topology/net2-lossy instproc init ns {
    $self instvar node_
    $self makenodes $ns
    $self createlinks $ns

    $self instvar lossylink_
    set lossylink_ [$ns link $node_(r1) $node_(r2)]
    set em [new ErrorModule Fid]
    set errmodel [new ErrorModel/Periodic]
    $errmodel unit pkt
    $lossylink_ errormodule $em
    $em insert $errmodel
    $em bind $errmodel 0
    $em default pass
}
```

```
TestSuite instproc finish file {
    global quiet PERL
    $self instvar ns_ tchan_ testName_ cwnd_chan_
        exec $PERL getrc -s 2 -d 3 all.tr | \
       $PERL raw2xg -a -e -s 0.01 -m 90 -t $file > temp.rands
    exec $PERL getrc -s 3 -d 2 all.tr | \
      $PERL raw2xg -a -e -s 0.01 -m 90 -t $file > temp1.rands
    if {$quiet == "false"} {
        exec xgraph -bb -tk -nl -m -x time -y packets

temp.rands &
# The line below plots both data and ack packets.
#           exec xgraph -bb -tk -nl -m -x time -y packets

temp.rands \
#                     temp1.rands &
    }
        ## now use default graphing tool to make a data file
        ## if so desired

    if { [info exists tchan_] && $quiet == "false" } {
        $self plotQueue $testName_
    }
    if { [info exists cwnd_chan_] && $quiet == "false" } {
        $self plot_cwnd
    }
    $ns_ halt
}

TestSuite instproc enable_tracequeue ns {
    $self instvar tchan_ node_
    set redq [[$ns link $node_(r1) $node_(r2)] queue]
    set tchan_ [open all.q w]
    $redq trace curq_
    $redq trace ave_
        #$redq monitor-queue {$node_(r1) $node_(r2)}
    $redq attach $tchan_


}
```

```
TestSuite instproc plotQueue file {
    global quiet
    $self instvar tchan_
    #
    # Plot the queue size and average queue size, for RED gateways.
    #
    set awkCode {
        {
            if ($1 == "Q" && NF>2) {
                print $2, $3 >> "temp.q";
                set end $2
            }
            else if ($1 == "a" && NF>2)
                print $2, $3 >> "temp.a";
        }
    }
    set f [open temp.queue w]
    puts $f "TitleText: $file"
    puts $f "Device: Postscript"

    if { [info exists tchan_] } {
        close $tchan_
    }
    exec rm -f temp.q temp.a
    exec touch temp.a temp.q

    exec awk $awkCode all.q

    puts $f \"queue
    exec cat temp.q >@ $f
    puts $f \n\"ave_queue
    exec cat temp.a >@ $f
    ###puts $f \n"thresh
    ###puts $f 0 [[ns link $r1 $r2] get thresh]
    ###puts $f $end [[ns link $r1 $r2] get thresh]
    close $f
    if {$quiet == "false"} {
        exec xgraph -bb -tk -x time -y queue temp.queue &
    }
}

TestSuite instproc tcpDumpAll { tcpSrc interval label } {
```

```
    global quiet
    $self instvar dump_inst_ ns_
    if ![info exists dump_inst_($tcpSrc)] {
    set dump_inst_($tcpSrc) 1
    set report $label/window=[$tcpSrc set

window_]/packetSize=[$tcpSrc set packetSize_]
    if {$quiet == "false"} {
        puts $report
    }
    $ns_ at 0.0 "$self tcpDumpAll $tcpSrc $interval $label"
    return
    }
    $ns_ at [expr [$ns_ now] + $interval] "$self tcpDumpAll $tcpSrc

$interval $label"
    set report time=[$ns_ now]/class=$label/ack=[$tcpSrc set

ack_]/packets_resent=[$tcpSrc set nrexmitpack_]
    if {$quiet == "false"} {
        puts $report
    }
}

TestSuite instproc emod {} {
    $self instvar topo_
    $topo_ instvar lossylink_
        set errmodule [$lossylink_ errormodule]
    return $errmodule
}

TestSuite instproc setloss {} {
    $self instvar topo_
    $topo_ instvar lossylink_
        set errmodule [$lossylink_ errormodule]
        set errmodel [$errmodule errormodels]
        if { [llength $errmodel] > 1 } {
                puts "droppedfin: confused by >1 err models..abort"
                exit 1
        }
    return $errmodel
}
```

```
TestSuite instproc setTopo {} {
    $self instvar node_ net_ ns_ topo_

    set topo_ [new Topology/$net_ $ns_]
    if {$net_ == "net2" || $net_ == "net2-lossy"} {
        set node_(s1) [$topo_ node? s1]
        set node_(s2) [$topo_ node? s2]
        set node_(s3) [$topo_ node? s3]
        set node_(s4) [$topo_ node? s4]
        set node_(r1) [$topo_ node? r1]
        set node_(r2) [$topo_ node? r2]
        [$ns_ link $node_(r1) $node_(r2)] trace-dynamics $ns_ stdout
    }
    if {$net_ == "net6"} {
        set node_(s1) [$topo_ node? s1]
        set node_(s2) [$topo_ node? s2]
        set node_(r1) [$topo_ node? r1]
        set node_(k1) [$topo_ node? k1]
        [$ns_ link $node_(r1) $node_(k1)] trace-dynamics $ns_ stdout
    }
}

###################################################################

Class Test/ecn -superclass TestSuite
Test/ecn instproc init {} {
    $self instvar net_ test_
    Queue/RED set setbit_ true
    set net_    net2
    set test_   ecn
    $self next
}
Test/ecn instproc run {} {
    $self instvar ns_ node_ testName_
    $self setTopo

    Agent/TCP set old_ecn_ 1
    set stoptime 10.0
    set redq [[$ns_ link $node_(r1) $node_(r2)] queue]
    $redq set setbit_ true
```

```
    set tcp1 [$ns_ create-connection TCP/Reno $node_(s1) TCPSink

$node_(s3) 0]
    $tcp1 set window_ 15
    $tcp1 set ecn_ 1

    set tcp2 [$ns_ create-connection TCP/Reno $node_(s2) TCPSink

$node_(s3) 1]
    $tcp2 set window_ 15
    $tcp2 set ecn_ 1

    set ftp1 [$tcp1 attach-app FTP]
    set ftp2 [$tcp2 attach-app FTP]

    $self enable_tracequeue $ns_
    $ns_ at 0.0 "$ftp1 start"
    $ns_ at 2.0 "$ftp2 start"

    $self tcpDump $tcp1 5.0

    # trace only the bottleneck link
    $self traceQueues $node_(r1) [$self openTrace $stoptime $testName_]

    $ns_ run

}

###################################################################

TestSuite instproc ecnsetup { tcptype {stoptime 3.0} { tcp1fid 0 } {

delack 0 }} {
    $self instvar ns_ node_ testName_ net_
    $self setTopo
##
##  Agent/TCP set maxburst_ 4
##
    set delay 280ms
    $ns_ delay $node_(r1) $node_(r2) $delay
    $ns_ delay $node_(r2) $node_(r1) $delay
```

```
    set redq [[$ns_ link $node_(r1) $node_(r2)] queue]
    $redq set setbit_ true

    if {$tcptype == "Tahoe" && $delack == 0} {
      set tcp1 [$ns_ create-connection TCP $node_(s1) \
      TCPSink $node_(s3) $tcp1fid]
    } elseif {$tcptype == "Sack1" && $delack == 0} {
      set tcp1 [$ns_ create-connection TCP/Sack1 $node_(s1) \
      TCPSink/Sack1  $node_(s3) $tcp1fid]
    } elseif {$delack == 0} {
      set tcp1 [$ns_ create-connection TCP/$tcptype $node_(s1) \
      TCPSink $node_(s3) $tcp1fid]
    } elseif {$tcptype == "Tahoe" && $delack == 1} {
      set tcp1 [$ns_ create-connection TCP $node_(s1) \
      TCPSink/DelAck $node_(s3) $tcp1fid]
    } elseif {$tcptype == "Sack1" && $delack == 1} {
      set tcp1 [$ns_ create-connection TCP/Sack1 $node_(s1) \
      TCPSink/Sack1/DelAck  $node_(s3) $tcp1fid]
    } else {
      set tcp1 [$ns_ create-connection TCP/$tcptype $node_(s1) \
      TCPSink/DelAck $node_(s3) $tcp1fid]
    }
    $tcp1 set window_ 30
    $tcp1 set packetsize_ 512
    $tcp1 set ecn_ 1
    set ftp1 [$tcp1 attach-source FTP]
    $self enable_tracecwnd $ns_ $tcp1

    #$self enable_tracequeue $ns_
    $ns_ at 0.0 "$ftp1 start"

    $self tcpDump $tcp1 5.0

    # trace only the bottleneck link
    $self traceQueues $node_(r1) [$self openTrace $stoptime $testName_]
}

TestSuite instproc second_tcp { tcptype starttime } {
    $self instvar ns_ node_
    if {$tcptype == "Tahoe"} {
      set tcp [$ns_ create-connection TCP $node_(s1) \
     TCPSink $node_(s3) 2]
```

```
    } elseif {$tcptype == "Sack1"} {
      set tcp [$ns_ create-connection TCP/Sack1 $node_(s1) \
          TCPSink/Sack1  $node_(s3) 2]
    } else {
      set tcp [$ns_ create-connection TCP/$tcptype $node_(s1) \
     TCPSink $node_(s3) 2]
    }
    $tcp set window_ 30
    $tcp set ecn_ 1
    set ftp [$tcp attach-app FTP]
    $ns_ at $starttime "$ftp start"
}

# Drop the specified packet.
TestSuite instproc drop_pkt { number } {
    $self instvar ns_ lossmodel
    set lossmodel [$self setloss]
    $lossmodel set offset_ $number
    $lossmodel set period_ 10000
}

TestSuite instproc drop_pkts pkts {
    $self instvar ns_ errmodel1
    set emod [$self emod]
    set errmodel1 [new ErrorModel/List]
    $errmodel1 droplist $pkts
    $emod insert $errmodel1
    $emod bind $errmodel1 1

}



##############################################################################
# Performace Tests #
##############################################################################

# Plain ECN
Class Test/ecn_nodrop_tahoe -superclass TestSuite
Test/ecn_nodrop_tahoe instproc init {} {
        $self instvar net_ test_
        Queue/RED set setbit_ true
```

```
        set net_    net2-lossy
    Agent/TCP set bugFix_ true
        set test_   ecn_nodrop_tahoe
        $self next
}
Test/ecn_nodrop_tahoe instproc run {} {
    $self instvar ns_
    Agent/TCP set old_ecn_ 1
    $self ecnsetup Tahoe 3.0
    $self drop_pkt 10000
    $ns_ run
}

# Two ECNs close together
Class Test/ecn_twoecn_tahoe -superclass TestSuite
Test/ecn_twoecn_tahoe instproc init {} {
        $self instvar net_ test_
        Queue/RED set setbit_ true
        set net_    net2-lossy
    Agent/TCP set bugFix_ true
        set test_   ecn_twoecn_tahoe
        $self next
}
Test/ecn_twoecn_tahoe instproc run {} {
    $self instvar ns_ lossmodel
    Agent/TCP set old_ecn_ 1
    $self ecnsetup Tahoe 5.0
    #$self drop_pkt 243
        $self drop_pkt 4
    #$lossmodel set markecn_ true

    $ns_ run
}

# ECN followed by packet loss.
Class Test/ecn_drop_tahoe -superclass TestSuite
Test/ecn_drop_tahoe instproc init {} {
        $self instvar net_ test_
        Queue/RED set setbit_ true
        set net_    net2-lossy
    Agent/TCP set bugFix_ true
        set test_   ecn_drop_tahoe
```

```
        $self next
}
Test/ecn_drop_tahoe instproc run {} {
    $self instvar ns_
    Agent/TCP set old_ecn_ 1
    $self ecnsetup Tahoe 3.0
    $self drop_pkt 243
    $ns_ run
}

# ECN preceded by packet loss.
Class Test/ecn_drop1_tahoe -superclass TestSuite
Test/ecn_drop1_tahoe instproc init {} {
        $self instvar net_ test_
        Queue/RED set setbit_ true
        set net_    net2-lossy
    Agent/TCP set bugFix_ true
        set test_   ecn_drop1_tahoe
        $self next
}
Test/ecn_drop1_tahoe instproc run {} {
    $self instvar ns_
    Agent/TCP set old_ecn_ 1
    $self ecnsetup Tahoe 3.0
    $self drop_pkt 241
    $ns_ run
}

# ECN preceded by packet loss.
Class Test/ecn_drop2_tahoe -superclass TestSuite
Test/ecn_drop2_tahoe instproc init {} {
        $self instvar net_ test_
        Queue/RED set setbit_ true
        set net_    net2-lossy
    Agent/TCP set bugFix_ true
        set test_   ecn_drop2_tahoe
        $self next
}
Test/ecn_drop2_tahoe instproc run {} {
    $self instvar ns_ node_ count_
    #Agent/TCP set old_ecn_ 1
    $self ecnsetup Tahoe 500.0 1
```

```
        $self drop_pkts {550 551}

    $ns_ run
    }

TestSuite runTest
```

Appendix C

TCL/TK CODES FOR TRACING SIMULATION RESULTS

```
Object instproc exit args {
      set ns [Simulator instance]
      catch "$ns clearTimers"
      eval exit $args
}

Class TestSuite

TestSuite instproc init { {dotrace 1} } {
    global quiet
    $self instvar ns_ test_ node_ testName_
    $self instvar allchan_ namchan_
    if [catch "$self get-simulator" ns_] {
        set ns_ [new Simulator]
    }
    if { $dotrace } {
                set allchan_ [open all.tr w]
                $ns_ trace-all $allchan_
        set namchan_ [open all.nam w]
        if {$quiet == "false"} {
                    $ns_ namtrace-all $namchan_
        }
    }
    set testName_ "$test_"
}
```

```
#
# Arrange for tcp source stats to be dumped for $tcpSrc every
# $interval seconds of simulation time
#
TestSuite instproc tcpDump { tcpSrc interval } {
    global quiet
    $self instvar dump_inst_ ns_
    if ![info exists dump_inst_($tcpSrc)] {
        set dump_inst_($tcpSrc) 1
        $ns_ at 0.0 "$self tcpDump $tcpSrc $interval"
        return
    }
    $ns_ at [expr [$ns_ now] + $interval] "$self tcpDump $tcpSrc

$interval"
    set report [$ns_ now]/cwnd=[format "%.4f" [$tcpSrc set

cwnd_]]/ssthresh=[$tcpSrc set ssthresh_]/ack=[$tcpSrc set ack_]
        if {$quiet == "false"} {
                puts $report
        }
}

#
# Trace the TCP congestion window cwnd_.
#
TestSuite instproc enable_tracecwnd { ns tcp } {
        $self instvar cwnd_chan_
        set cwnd_chan_ [open all.cwnd w]
        $tcp trace cwnd_
        $tcp attach $cwnd_chan_
}

#
# Plot the TCP congestion window cwnd_.
#
TestSuite instproc plot_cwnd {} {
        global quiet
        $self instvar cwnd_chan_
        set awkCode {
              {
              if ($6 == "cwnd_") {
```

```
                print $1, $7 >> "temp.cwnd";
              } }
        }
        set f [open cwnd.xgr w]
        puts $f "TitleText: cwnd"
        puts $f "Device: Postscript"

        if { [info exists cwnd_chan_] } {
                close $cwnd_chan_
        }
        exec rm -f temp.cwnd
        exec touch temp.cwnd

        exec awk $awkCode all.cwnd

        puts $f \"cwnd
        exec cat temp.cwnd >@ $f
        close $f
        if {$quiet == "false"} {
                exec xgraph -bb -tk -x time -y cwnd cwnd.xgr &
        }
}

TestSuite instproc cleanup { tfile testname } {
    $self instvar ns_ allchan_ namchan_
    $ns_ halt
    close $tfile
    if { [info exists allchan_] } {
        close $allchan_
    }
    if { [info exists namchan_] } {
        close $namchan_
    }
    $self finish $testname; # calls finish procedure in test suite

file
}


TestSuite instproc openTrace { stopTime testName } {
    $self instvar ns_ allchan_ namchan_
    exec rm -f out.tr temp.rands
```

```
    set traceFile [open out.tr w]
    puts $traceFile "v testName $testName"
    $ns_ at $stopTime "$self cleanup $traceFile $testName"
    return $traceFile
}

TestSuite instproc traceQueues { node traceFile } {
        $self instvar ns_
        foreach nbr [$node neighbors] {
                $ns_ trace-queue $node $nbr $traceFile
                [$ns_ link $node $nbr] trace-dynamics $ns_ $traceFile
        }
}

proc usage {} {
    global argv0
    puts stderr "Valid tests are:\t[get-subclasses TestSuite

Test/]"
    exit 1
}

proc isProc? {cls prc} {
    if [catch "Object info subclass $cls/$prc" r] {
        global argv0
        puts stderr "$argv0: no such $cls: $prc"
        usage
    }
}

proc get-subclasses {cls pfx} {
    set ret ""
    set l [string length $pfx]

    set c $cls
    while {[llength $c] > 0} {
        set t [lindex $c 0]
        set c [lrange $c 1 end]
        if [string match ${pfx}* $t] {
            lappend ret [string range $t $l end]
        }
        eval lappend c [$t info subclass]
```

```
    }
    set ret
}

TestSuite proc runTest {} {
        global argc argv quiet

        set quiet false
        switch $argc {
                1 {
                        set test $argv
                        isProc? Test $test
                }
                2 {
                        set test [lindex $argv 0]
                        isProc? Test $test

                        set param [lindex $argv 1]
                        if {$param == "QUIET"} {
                                set quiet true
                        }
                }
                default {
                        usage
                }
        }
        set t [new Test/$test]
        $t run
}

### Local Variables:
### mode: tcl
### tcl-indent-level: 8
### tcl-default-application: ns
### End:
```

Appendix D

C++ CODES OF DIFF-C-TCP

Available upon request; To save space here.