



# **Modeling and Analysis of TCP Reno with Packet-Loss and Long Delay Cycles in Wireless Networks**

نمذجة وتحليل بروتوكول التحكم في معدل ارسال البيانات وفقا لنظام رينو مع فقدان  
حزمة البيانات ومدة التأخير في دورة ارسال البيانات في الشبكات اللاسلكية

**By**

**Haitham Y. Adarbah**

**(20051240887)**

**Supervisor**

**Dr. Hussein Ismail Al-Bahadili**

This thesis is submitted to the Department of Computer Science,  
Graduate College of Computing Studies, Amman Arab University for  
Graduate Studies in partial fulfillment of the requirement for the degree  
of Master of Science in Computer Science.

**Graduate College of Computing Studies  
Amman Arab University for Graduate Studies**

**(August – 2008)**

### Authorization of Dissemination

I the undersigned " Haitham Yousef Adarbah" authorize hereby Amman Arab University for Graduate Studies to provide copies of this thesis to libraries, institution and any other parties upon their request.

Name : Haitham Yousef Adarbah

Signature:

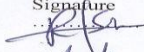
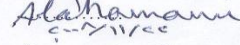
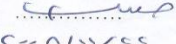


Date: 29/11/2008

### The Discussion Committee Decision

The dissertation entitled " Modeling and Analysis of TCP Reno with Packet-Loss and Long Delay Cycles in Wireless Networks" was discussed and passed on November, 3, 2008.

#### Discussion Committee Members

		Signature
Dr. Riyadh Al- Shalabi	Chair man	
Prof. Dr. Ala'a Al-Hamami	Member	 Ala'hamami علاء حمّامى
Dr. Hussein Al-Bahadili	Member and supervisor	 عبد الحسين البهادلي

## Abstract

Modeling and simulations have been widely used to evaluate the performance of the Transport Control Protocol (TCP) in a wireless environment. However, for many reasons, several efforts have been also directed to model and evaluate the performance of TCP in a wireless environment. Therefore, during the last two decades, many models have been developed. But, they all have their own limitations, due to the approximation they assumed during the modeling process, e.g., effects of long delays, frequency of occurrence of long delays, Slow-Start stage, Acknowledgement (ACK) mechanism, etc.

The main objective of this work is to develop a new model that can be used to evaluate the performance of TCP Reno in a wireless environment that suffers from Packet-Loss (PL) and Long Delay Cycles (LDCs), namely, the PLLDC model. It models the TCP Reno Sending rate ( $S$ ), Throughput ( $T$ ), and Utilization factor ( $U$ ) as a function of environment- and system-driven parameters. The former include: Packet-Loss rate ( $p$ ), duration of the Long Delay ( $D$ ), Interval between Long Delays ( $I$ ), and Round Trip Time ( $RTT$ ), while the latter include: Timeout ( $T_o$ ), Slow-Start Threshold at the end of a Long Delay ( $SST$ ), number of packets acknowledged by one ACK packet ( $b$ ), and the receiver's maximum congestion Window size ( $W_m$ ).

In the PLLDC model, an LDC is assumed to consist of  $m$  instances of normal points (NPs) at the beginning and Long Delay Period (LDP) at the end. The duration of NP is the sum of the duration of  $n$  Triple Duplicate Periods (TDPs) and Timeout period. Furthermore, the duration of one LDP consists of two TDP periods, Long Delay, and one Slow-Start stage, minus the overlapping period between duration of the Long Delay and TDP.

The main advantage of the PLLDC model over other existing models is that it can be used to accurately predict the number of packets sent during all of the above durations, so that a detailed assessment of the TCP performance can be achieved, and proper solutions to enhance TCP may be proposed.

To validate the PLLDC model, it is used to estimate  $S$ ,  $T$ , and  $U$  for various input parameters in three different scenarios. The results obtained are validated against equivalent simulation results obtained by NS-2 network simulator, and those determined by the well-known PFTK model. Our model provides more accurate results than the PFTK model. The results obtained are discussed and presented in tables and graphs. Finally, conclusions are drawn, and recommendations for future work are pointed-out.

## Arabic Summary

### ملخص

النماذج والمحاكاة أصبحت الأكثر انتشارا لقياس أداء بروتوكول التحكم في ارسال البيانات وهناك عدة عوامل تؤثر بصورة مباشرة على عمل النماذج وأداء بروتوكول التحكم في معدل ارسال البيانات ضمن بيئة الشبكات اللاسلكية.

وخلال السنوات الماضية تم تطوير عدة نماذج لقياس أداء عمل بروتوكول التحكم في معدل ارسال البيانات، وكلها كانت محدودة في قياسها لعدم احتوائها على عدد من العوامل منها مدة التأخير في ارسال البيانات وتكرار مدة التأخير، البدء البطيء، تأكيد على الوصول الخ. الهدف الرئيسي في هذا العمل هو تطوير نموذج جديد يستخدم لقياس أداء التحكم في معدل ارسال البيانات وفقا لنظام رينو ضمن بيئة الشبكات اللاسلكية مع وجود فقدان لحزمة البيانات ورمز لها (PL) ومدة التأخير في دورة ارسال البيانات ورمز لها (LDC) وتم تسمية النموذج بـ (PLLDC MODLE).

وكما انه يقيس معدل ارسال البيانات (S)، و معدل الإرسال الحقيقي (الإنتاجية) (T)، ونسبة معدل الإرسال للبيانات على معدل الإرسال الحقيقي (الإنتاجية) ورمز لها بـ (U) ويشمل النموذج على المتغيرات التالية :

(p) نسبة فقدان حزم البيانات.

(D) فترة مدة التأخير في ارسال البيانات.

(I) الفترة الزمنية التي يتم فيها تكرار حدوث مدة التأخير.

(RTT) رحلة الذهاب والإياب في عملية الإرسال.

(T<sub>0</sub>) الوقت المنتهي في عملية الإرسال.

(SST) عتبة البدء البطيء.

(b) عدد حزم البيانات التي تم ارسالها والتأكيد على وصولها.

( $W_m$ ) أكبر عدد ممكن يتم فيه ارسال حزم بيانات دون حدوث تصادم أو ازدحام. نموذج ( PLLDC MODEL ) يتكون من ( $m$ ) في معدل ارسال طبيعي (NP) في البداية ثم فترة مدة التأخير (LDP) في النهاية. حيث أن:

(NP): تتكون من ( $n$ ) في الفترة التي تسمى (TDP) وفترة نهاية الوقت (Timeout).

(LDP): تتضمن فترتين من (TDP) ومدة تأخير و البدء البطيئ (Slow-Star)، ناقصا التداخل بين فترة التأخير و فترة (TDP).

الفائدة المهمة في النموذج (PLLDC Model) هي إعطاء قيم أكثر دقة لمعدل ارسال البيانات وأيضا المساعدة على إعطاء حلول لتحسين أداء بروتوكول التحكم في معدل ارسال البيانات. لعملية التحقق من صحة النموذج (PLLDC Model) تم قياس  $S, T, U$  عند ادخال قيم مختلفة للمتغيرات في ثلاث مواقف، وتم التأكد من من خلال مقارنتها بنتائج محاكاة الشبكات المشهور NS-2 والنموذج (PFTK Model)، حيث اظهرت النتائج أن القيم التي تم الحصول عليها من (PLLDC Model) أكثر دقة وأقرب الى الواقع، وتم عرض النتائج على شكل جداول و رسومات بيانية، و في النهاية تم وضع المساهمات التي يمكن أن يساهم بها هذا النموذج والتوصيات للمستقبل حول النموذج.

## Acknowledgment

First, I would like to express my deep appreciation and thanks to my thesis supervisor Dr. Hussein Al-Bahadili, for his guidance during each stage of this research, for answering endless questions, for always being willing to spare his time to resolve my queries. I would especially like to thank him for his extensive comments on the thesis write up, as he has also helped me to present my research in a much better way. Always his encouragement was a great source of motivation for me.

Second, I would like to express my deep appreciation and thanks to Mr. Ghaeth Hafar, for his guidance my during undergraduate study, I would especially like to thank him for his extensive comments on my life, as he has also helped me to shape my future life in a much better way.

Third, it would be unthinkable of me not to thank my wonderful wife for always encouraging me to pursue my beliefs, my ideas and my dreams. Her faithful and loyal support has been instrumental in achieving my goal. Also I thank her for her unwavering support, motivation and invaluable help in finishing this thesis.

I dedicate this thesis in memory of my mother. Also I thank her for her unwavering support, motivation and invaluable help in my all life.



# Table of Contents

Abstract.....	IV
Arabic Summary .....	VI
Acknowledgment.....	VIII
Table of Contents.....	IX
List of Figures .....	XII
List of Tables.....	XIV
Abbreviations .....	XV
Chapter 1 Introduction .....	1
1.1. Transport Control Protocol (TCP) .....	1
1.1.1 TCP header format .....	3
1.1.2. TCP connection .....	5
1.2 Congestion of TCP.....	6
1.3 TCP Reno .....	12
1.4 Performance of TCP .....	13
1.5 Modeling of TCP .....	15
1.5.1 Essential of TCP modeling.....	15
i. Window dynamics .....	16
ii. Packet-Loss process.....	17
1.6 Motivation.....	18
1.7 Statement of the Problem .....	19
1.8 Thesis Organization .....	20
Chapter 2 Literature Review .....	21
2.1 Introduction .....	21

2.2	TCP Literature Review .....	22
Chapter 3 The Packet-Loss and Long Delay Cycles (PLLDC) Model .....		32
3.1	The PFTK Congestion Control Model .....	33
3.1.1	Features of the PFTK model .....	33
3.1.2	Modeling of the TCP Reno sending rate ( $S$ ) (PFTK model) .....	35
3.1.3	Modeling of the TCP Reno throughput ( $T$ ) (PFTK model) .....	43
3.2	The PLLDC Modeling Assumptions .....	44
3.3	Basics of the PLLDC TCP Analytical Model .....	45
3.3.1	Dynamics of sender window around a long delay .....	45
3.3.2	Statistical modeling of the long delay pattern .....	48
3.4	Modeling of the TCP Reno Sending Rate ( $S$ ) (PLLDC Model) .....	49
3.4.1	Analysis of a long delay period (LDP) .....	50
3.4.2	Analysis of one long delay cycle (LDC) .....	52
3.5	Modeling of the TCP Reno Throughput ( $T$ ) (PLLDC Model) .....	55
3.6	Modeling of the TCP Reno Utilization Factor ( $U$ ) (PLLDC Model) .....	58
3.7	Implementations .....	59
Chapter 4 Results and Discussions .....		62
4.1	Scenario #1: Investigates the Effect of Long Delays ( $D$ ) .....	64
4.1.1	Scenario #1: Sending rate ( $S$ ) .....	64
4.1.2	Scenario #1: Throughput ( $T$ ) .....	69
4.1.3	Scenario #1: Utilization factor ( $U$ ) .....	73
4.2	Scenario #2: Investigates the Effect of Intervals between Long Delays ( $I$ ) 74	
4.2.1	Scenario #2: Sending rate ( $S$ ) .....	74
4.2.2	Scenario #2: Throughput ( $T$ ) .....	76
4.2.3	Scenario #2: Utilization factor ( $U$ ) .....	78

4.3 Scenario #3: Investigates the Effect of Round Trip Time ( <i>RTT</i> ).....	79
4.3.1 Scenario #3: Sending rate ( <i>S</i> ).....	79
4.3.2 Scenario #3: Throughput ( <i>T</i> ).....	81
4.3.3. Scenario #3: Utilization factor ( <i>U</i> ) .....	83
Chapter 5 Conclusions and Recommendations for Future Work .....	85
5.1 Conclusions .....	85
5.2 recommendations for Future Work.....	86
Reference .....	88

# List of Figures

<u>Figure</u>	<u>Title</u>	<u>Page</u>
1.1	TCP segment format	2
3.1	Evolution of window size over time when loss indications are TD.	34
3.2	Packets sent during a TDP	36
3.3	Evolution of window size when loss indications are TD and TO.	37
3.4	Evolution of window size limited by $W_m$ .	39
3.5	Fast retransmit with window limitation	40
3.6	Packet sent during one long delay period (LDP)	44
3.7	(a) Variation of $RTT$ showing four long delays, and (b) model og long delays	44
3.8	Sender window evaluation in one LDC	51
4.1	Variation of $S$ with $p$ for various values of $D$ for Scenario #1	63
4.2	Comparison of $S$ for Scenario #1.	64
4.3	Packets send during the different stages of a Long Delay Cycle for various values of $D$ for Scenario #1	66
4.4	Comparison of $T$ for Scenario #1	69
4.5	Variation of $T$ with $p$ for various values of $D$ for Scenario #1.	70
4.6	Variation of $U$ with $p$ for various values of $D$ for Scenario #1.	71
4.7	Variation of $S$ with $p$ for various values of $I$ for Scenario #2	74

4.8	Comparison of $T$ for Scenario #2.	75
4.9	Variation of $U$ with $p$ for various values of $I$ for Scenario #2.	76
4.10	Variation of $S$ with $p$ for various values of $RTT$ for Scenario #3.	79
4.11	Variation of $T$ with $p$ for various values of $RTT$ for Scenario #3.	81
4.12	Variation of $U$ with $p$ for various values of $RTT$ for Scenario #3.	82

## List of Tables

<u>Table</u>	<u>Title</u>	<u>Page</u>
3.1	Definition of the code input parameters.	57
3.2	Definition of the code computed parameters	57
4.1	Input Parameter for Scenario #1	61
4.2	Comparison of the sending rate (S) for Scenario #1	62
4.3	Number of packets sent during the different time periods of a Long Delay Cycle.	65
4.4	Comparison of the throughput (T) for Scenarion #1	68
4.5	Input Parameter for Scenario #2	72
4.6	Comparison of the sending rate (S) for Scenario #2	73
4.7	Comparison of the throughput (T) for Scenarion #2	75
4.8	Input Parameter for Scenario #3	77
4.9	Comparison of the sending rate (S) for Scenario #2	78
4.10	Comparison of the throughput (T) for Scenarion #2	78

## Abbreviations

ACK	Acknowledgement.
AIMD	Additive Increase Multiplicative Decrease.
AP	Access Point.
CDMA	Code Division Multiple Access
Cwnd	Congestion Window size
DSACKs	Duplicate Selective Acknowledgement
DUPACK	Duplicate Acknowledgement
FTP	File Transfer Protocol
GPRS	General Packet Radio Service
HMIP	Hierarchical Mobile IP
HTTP	Hypertext Transfer Protocol
IP	Internet Protocol
LDC	Long Delay Cycle
LDP	Long Delay Period
LIMD	Linear Increase Multiplicative Decrease
LLC	Link Layer Control

MAC	Media Access Control
MANETs	Mobile Ad hoc Networks
MIMD	Multiplicative Increase Multiplicative Decrease
MIP-RO	Mobile IP Route Optimaization
MSS	Maximum Segment Size
MTU	Maximum Transfer Unit
NP	Normal Period
NS	Network Simulator
PDF	Probability Distribution Function
PFTK	Initial of authors
PL	Packet Loss
QoS	Quality of Service
RLC	Radio Link Control
RTO	Retransmission timeout
RTT	Round Trip Time
SACK	Selective Acknowledgment
SCTP	Stream Control Transmission Protocol.



SDC	Simpl Delay Control
SFR	Suprious Fast Retransmission
SR	Suprious Retransmission
SS	Slow Start
SST	Slow Start threshold
ST	Suprious Timeout
TCP	Transmission Control Protocol
TD	Triple Duplicate
TDP	Triple Duplicate Period
TFRC	TCP-Friendly Rate Contorl.
TO	Timeout
TSN	Transmission Sequence Number
WLAN	Wireless Local Area Network
WMAN	Wireless Metroplatn Area Network

# Chapter 1

## Introduction

### 1.1. Transport Control Protocol (TCP)

The Transport Control Protocol (TCP) is the dominant transport layer protocol in the Internet Protocol (IP) suite. It carries a significant amount of the Internet traffics, such as Web browsing, files transfer, e-mail, and remote access. It is a reliable connection-oriented protocol that allows a byte stream originating on one machine to be delivered without error to any other machine in the Internet. It fragments the incoming byte stream into discrete messages, not exceeding 64 KB, and passes each one on to the network layer. At the destination, the receiving TCP process reassembles the received messages into the output stream [For 07, Sta 08, Tan 03].

An Internet work differs from a single network because different parts may have different topologies, delays, bandwidths, packet sizes, and other parameters. TCP was designed to meet the following main objectives [Vio 07]:

- Dynamically adaptable to outfit internet work.
- Robust in the face of many kinds of failures.
- Handle flow control to make sure a fast sender cannot swamp a slow receiver with more messages than it can handle.
- Support full duplex and point-to-point connections.

But, it does not support multicasting or broadcasting. A key feature of TCP is that every byte on a TCP connection has its own 32-bit sequence number. The sending and receiving TCP entities exchange data in the form of segments. Figure (1.1) shows the TCP segment format. A TCP segment consists of a fixed 20-byte header (plus an optional part) followed by zero or more data bytes. The TCP software decides how big segments should be, and can accumulate data from several writes into one segment or can split data from one write over multiple segments.

Two limits restrict the segment size [Com 06]:

- Each segment, including the TCP header, must fit in the 65,515-byte IP payload.
- Each network has a Maximum Transfer Unit (MTU), and each segment must fit in the MTU.

In practice, the MTU is generally 1500 bytes (the Ethernet payload size) and thus defines the upper bound on segment size. The basic protocol used by TCP entities is the sliding window protocol. Segments can arrive out of order, so bytes 3072-4095 can arrive but cannot be acknowledged because bytes 2048-3071 have not turned up yet. Segments can also be delayed so long in transit that the sender times out and retransmits them. The retransmissions may include different byte ranges than the original transmission. TCP must be prepared to deal with these problems and solve them in an efficient way.

The definition of the components of the TCP segment header is given below to provide the reader with some information that assists in understanding the modelling process, and more details can be found in many computer networks textbooks and literatures [Tan 03].

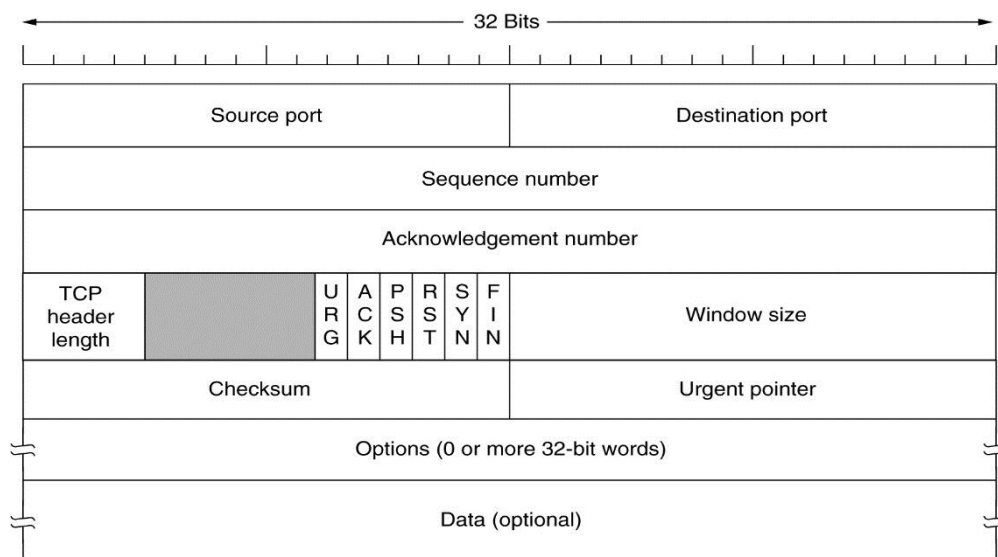


Figure (1.1). TCP segment format [Tan 03].

## 1.1.1 TCP header format

The TCP segment has two parts, a standard 20-byte header followed by a variable payload containing the application data, as shown in Figure (1.1). The header contains much useful information, such as the advertised window size, ACK number, and so on. In what follows a description is given for the fields in the TCP header and their meanings [For 07, Com 06, and Tan 03].

- **Source Port (16-bit)**. Source end-point of the connection.
- **Destination Port (16-bit)**. Destination end-point of the connection.
- **Sequence Number (32-bit)**. It contains the sequence number of the first byte of data carried in the TCP segment. Last byte correctly received. As an example, if the preceding segment started with a sequence number of 2001 and contained 1460 bytes of data, then the sequence number of the next TCP segment is set to 3461.
- **Acknowledgement Number (32-bit)**. The destination uses this field to acknowledge the correctly received data, by putting in the next byte expected.
- **Header Length (4-bit)**. It indicated the length of the TCP header in multiples of 32-bit words. In most cases the header length of a TCP segment is 20 bytes; however, this may vary if the option field is used. Because the header can be of variable length, the length field also helps to identify the start of the payload.
- **Future Development (6-bit)**. These six bits are reserved for future or experimental use.
- **Flags (6-bit)**. A TCP segment may carry several different types of protocol messages, such as ACK, start signal of a connection, end signal of a connection, and so on. Each bit in the flag field is used to identify a given type.

- **URG (U) (1-bit)**. Urgent pointer field valid.
- **ACK (A) (1-bit)**. It sets to 1 to indicate that the Acknowledgement number is valid. If ACK is 0, the segment does not contain an acknowledgement so the Acknowledgement number field is ignored.
- **PSH (P) (1-bit)**. It indicates PUSHed data. The receiver is hereby kindly requested to deliver the data to the application upon arrival and not buffer it until a full buffer has been received.
- **RST (R) (1-bit)**. It is used to reset a connection that has become confused due to some reasons. It is also used to reject an invalid segment or refuse an attempt to open a connection.
- **SYN (S) (1-bit)**. It is used to establish connections. The connection request has SYN=1 and ACK=0 to indicate that the piggyback acknowledgement field is not in use. The connection reply does bear an acknowledgement, so it has SYN=1 and ACK=1. In essence the SYN bit is used to denote CONNECTION REQUEST and CONNECTION ACCEPTED, with the ACK bit used to distinguish between those two possibilities.
- **FIN (F) (1-bit)**. It is used to release a connection. It specifies that the sender has no more data to transmit. After closing a connection, the closing process may continue to receive data indefinitely. Both SYN and FIN segments have sequence numbers and are guaranteed to be processed in the correct order.
- **Receiver Window Size (16-bit)**. It tells how many bytes may be sent starting at the byte acknowledged. A Window size field of 0 is legal and says that the bytes up to and including Acknowledgement number-1 have been received, but the receiver would like no more data for the moment.

- **Checksum (16-bit)**. The checksum field is computed over the TCP header, the TCP payload, and the pseudoheader consisting of the source and destination IP addresses as well as the length field of the IP header. It provides extra reliability.
- **Urgent Pointer (16-bit)**. A TCP segment may carry data that need prior treatment (the urgent URG flag would be set for this segment).
- **Options (Variable)**. Options are to be specified using multiples of four bytes. There are two extra bytes preceding each option. The first byte indicates the option type followed the second byte indicating the length of the option in bytes (including these two preceding bytes). Examples of options are:
  - ***Maximum segment size (MSS) (16-bit)***. This option is used by the originating TCP during connection establishment (in the start of a new connection (SYN) segment) to negotiate the MSS to be used for the connection. The 16 bits used for this field limit the MSS to 64 KB.
  - ***Timestamps (64-bit)***. The timestamp option is to be used for more accurate round-trip time (RTT) calculations. Two four-byte timestamp fields are used for this option. The sending TCP fills the first field with the current time. The receiver echoes back the timestamp value received in the second field in an ACK segment. This facilitates the sender for more accurate calculation of the RTT.

### 1.1.2. TCP connection

TCP connection consists of three main components, these are: TCP sender, TCP link, and TCP receiver. The TCP sending rate is derived by the capacity and reliability of the TCP link and receiver. More details on the behavior of the TCP sender and link will be given in the next sections, while the behavior of the TCP receiver is given below.

Whenever a TCP receives data, it incurs some sort of obligation to generate an acknowledgement (ACK) in response to that data. The obligation may be optional or mandatory. An optional ACK obligation refers to data that the TCP may choose to acknowledge but can also wait before acknowledging. In other words, when new data arrives, the TCP standard states that a TCP may refrain from acknowledging such data in the hopes that additional data may arrive and the ACKs combined, but for no longer than 500 msec.

A TCP receiver sends two types of ACKs. It sends positive ACKs for segments that are received correctly and in-order, and it sends duplicate ACKs (DUPACKs) for segments that are received correctly but out-of-order. A DUPACK acknowledges the same sequence number that the last sent ACK acknowledged. Thus, a DUPACK does not convey which segment was received correctly [Che 08, Bla 04].

## 1.2 Congestion of TCP

TCP is the dominant transport protocol in the internet, and the current stability of the internet depends on its end-to-end congestion control. Therefore, applications sharing a best-effort network need to positively respond to congestion to ensure network stability and high performance. Traditionally, congestion control algorithms have been implemented at the transport layer; therefore, it is referred to as congestion of TCP. One of the key elements for any TCP congestion control algorithm is the congestion signal that informs senders that congestion has or is about to occur. There is no explicit way that can be adopted by a TCP source (sender) for congestion signal detection [Tek 08, Les 07, Wei 05, and Ven 03].

Two implicit approaches have been identified for congestion signal detection, these are:

- i. Loss-based approach.
- ii. Delay-based approach.

It is often not possible to draw sound conclusions on congestion from network delay measurements. Because it is difficult to find characteristic measures, such as the path's minimum RTT, due to persistent congestion at the bottleneck link or because of route changes.

Consequently, Packet-Loss is the only signal that senders can confidently use as an indication of congestion. A perceived packet-loss is implemented either as a direct or an indirect trigger to throttle the flow's send rate; such flows are referred to as loss responsive. In this sense, a TCP-based flow is a reliable loss responsive flow.



One disadvantage of Packet-Loss is that it is not unmistakable. Packets can get lost because of packet drops due to a buffer overflow at the bottleneck link or because of packet corruption due to a transmission error. The former indicates congestion, the latter does not. A sender is not able to discriminate among these events, because packet corruption usually leads to a frame checksum error and subsequent discard of the packet at the link layer.

Hence, transmission errors inevitably lead to an underestimation of available bandwidth for loss responsive flows. As a consequence, applications can only fully utilize their share of bandwidth along the path if transmission errors are rare events. Due to the high error rate in wireless links, wireless links are often problematic, and the Packet-Loss process and its consequences can not be safely neglected as in wire line links.

Two types of windows can be identified in a TCP connection, the congestion and advertised windows. The congestion window determines the number of bytes that can be outstanding at any time, or the maximum number of bytes can be transmitted without ACK that being received. This is a means of stopping the link between two places from getting overloaded with too much traffic. The size of this window is calculated by estimating how much congestion there is between the two places. Basically the size of the window, to a large degree, controls the speed of transmission as transmission pauses until there is ACK. The advertised window determines the number of bytes than can be sent over the TCP connection, which is imposed by the receiver. It is related to the amount of available buffer space at the receiver for this connection.

TCP congestion control consists of four mechanisms [Ho 08, Voi 07, Com 06, alt 05]:

- (1) Additive Increase/Multiplicative Decrease (AIMD).
- (2) Slow-Start
- (3) Fast retransmit
- (4) Fast recovery

In what follows a brief description is given for each of the above mechanisms.

(1) Additive Increase/Multiplicative Decrease (AIMD).

This algorithm is a feedback control algorithm used in TCP congestion avoidance. Basically, AIMD represents a linear growth of the congestion window, combined to an exponential reduction when congestion takes place. The approach taken is to increase the transmission rate (window size), probing for usable bandwidth, until loss occurs. The policy of additive increase is basically to increase the congestion window by 1 Maximum Segment Size (MSS) every RTT until a loss is detected. When loss is detected, the policy is changed to be one of multiplicative decrease which is to cut the congestion window in half after loss. The result is a saw tooth behavior that represents the probe for bandwidth.

(2) Slow-Start

Slow-Start is part of the congestion control strategy used by TCP in many Internet applications, such as HTTP, it is also known as the exponential growth phase. Slow-Start is used in conjunction with other algorithms to avoid sending more data than the network is capable of transmitting, that is, network congestion.

**The basic Slow-Start** algorithm begins in the exponential growth phase initially with a congestion window size (*cwnd*) of 1 or 2 segments and increases it by 1 segment size (*SS*) for each ACK received. This behavior effectively doubles the window size each round trip of the network. This behavior continues until the *cwnd* reaches the size of the receivers advertised window or until a loss occurs.

When a loss occurs half of the current *cwnd* is saved as a Slow-Start Threshold (*SSThresh*) and Slow-Start begins again from its initial *cwnd*. Once the *cwnd* reaches the *SSThresh* TCP goes into congestion avoidance mode where each ACK increases the *cwnd* by  $SS * SS / cwnd$ . This results in a linear increase of the *cwnd*.

### (3) Fast retransmit

Modifications to the congestion avoidance algorithm were proposed in 1990. Before describing the change, it is necessary realize that TCP may generate an immediate ACK or a DUPACK, when an out of order segment is received. This DUPACK should not be delayed. The purpose of this DUPACK is to let the other end knows that a segment was received out of order, and to tell it what sequence number is expected.

Since TCP does not know whether a DUPACK is caused by a lost segment or just a reordering of segments, it waits for a small number of DUPACKs to be received. It is assumed that if there is just a reordering of the segments, there will be only one or two DUPACKs before the reordered segment is processed, which will then generate a new ACK. If three or more DUPACKs are received in a row, it is a strong indication that a segment has been lost. TCP then performs a retransmission of what appears to be the missing segment, without waiting for a retransmission timer to expire.

### (4) Fast recovery

After fast retransmit sends what appears to be the missing segment, congestion avoidance, but not Slow-Start is performed. This is the fast recovery algorithm. It is an improvement that allows high throughput under moderate congestion, especially for large windows. The reason for not performing Slow-Start in this case is that the receipt of the DUPACKs tells TCP more than just a packet has been lost. Since the receiver can only generate the DUPACK when another segment is received, that segment has left the network and is in the receiver's buffer. In other words, there is still data flowing between the two ends, and TCP does not want to reduce the flow abruptly by going into Slow-Start.

The fast retransmit and fast recovery algorithms are usually implemented together as follows.

1. When the third DUPACK in a row is received, set *SSThresh* to one-half the current *cwnd* but no less than two segments. Retransmit the missing segment. Set *cwnd* to *SSThresh* plus 3 times the segment size. This inflates the congestion window by the number of segments that have left the network and which the other end has cached.
2. Each time another DUPACK arrives, increment *cwnd* by the segment size. This inflates the congestion window for the additional segment that has left the network. Transmit a packet, if allowed by the new value of *cwnd*.
3. When the next ACK arrives that acknowledges new data, set *cwnd* to *ssthresh* (the value set in step 1). This ACK should be the ACK of the retransmission from step 1, one RTT after the retransmission. Additionally, this ACK should acknowledge all the intermediate segments sent between the lost packet and the receipt of the first DUPACK. This step is congestion avoidance, since TCP is down to one-half the rate it was at when the packet was lost.

The fast retransmit algorithm first appeared in the Tahoe release, and it was followed by slow-start. The fast recovery algorithm appeared in the Reno release. Since, in this work we are concerned with TCP Reno, a description of this TCP flavor is given in the next section.

## 1.3 TCP Reno

The TCP Reno implementation retained the enhancements incorporated into Tahoe, but modified the fast retransmit operation to include fast recovery. The new algorithm prevents the communication path from going empty after fast retransmit, thereby avoiding the need to slow-start to refill it after a single packet-loss [Lul 04, Kam 03].

Fast recovery operates by assuming each DUPACK received represents a single packet having left the communication path. Thus, during fast recovery the TCP sender is able to make intelligent estimates of the amount of outstanding data. Fast recovery is entered by a TCP sender after receiving an initial threshold of DUPACKs. Once the threshold of DUPACKs is received, the sender retransmits one packet and reduces its congestion window by one half. Instead of slow-starting, as is performed by a Tahoe TCP sender, the Reno sender uses additional incoming DUPACKs to clock subsequent outgoing packets.

During fast recovery the sender “inflates” its window by the number DUPACKs it has received, according to the observation that each DUPACK indicates some ACK has been removed from the network and is now cached at the receiver. After entering fast recovery and retransmitting a single packet, the sender effectively waits until half a window of DUPACKs have been received, and then sends a new packet for each additional DUPACK that is received.

Upon receipt of an ACK for new data (called a “recovery ACK”), the sender exits fast recovery. Reno's fast recovery algorithm is optimized for the case when a single packet is dropped from a window of data. The Reno sender retransmits at most one dropped packet per RTT. Reno significantly improves upon the behavior of Tahoe TCP when a single packet is dropped from a window of data, but can suffer from performance problems when multiple packets are dropped from a window of data [Che 08, Xin 06, Lai 02].

## 1.4 Performance of TCP

A wireless network may suffer from extensive data loss due to: transmission errors in noisy environment, non-reliable wireless communication links, variable capacity links, frequent disconnections, limited communication bandwidth, broadcast nature of the communications, etc. Therefore, a wireless data communication session may involve a lot of data retransmission that degrades the performance of the networks. Data retransmissions reduce bandwidth utilization, and on the other hand increase delay and power consumption. These retransmissions are unavoidable and, in this thesis, we referred to them as factual retransmissions, because the data is lost and it will not reach the destination and they have to be retransmitted [Fu 03].

There is another form of data retransmission that is initiated by the TCP sender, which is referred to it as spurious retransmissions (SR) or spurious fast retransmission (SFR), which occurred when segments get re-ordered beyond the DUPACK-threshold in the network before reaching the receiver, i.e. the reordering length is greater than the DUPACK threshold (three for TCP). There are two main reasons for SFR, these are:

1. Timeout-based retransmission.
2. DUPACK-based retransmission.

DUPACK-based retransmission is triggered when three successive (triple) DUPACKs for the same sequence number have been received, i.e., without waiting for the retransmission timer to expire.

Timeout-based retransmission can be explained as follows. Since, TCP was initially designed for wired networks, and hence performs poorly in the presence of delay spikes which are especially more frequent in wireless networks than in traditional wired network [Gur 01a, Kha 02, Yav 02, Eom 02, Gur 02].

These delay spikes may exhibit a sudden increase in the instantaneous RTT beyond the sender's retransmission timeout (RTO) (or simply abbreviated as TO) value causes retransmission ambiguity, resulting in spurious timeout (ST), which is defined as a timeout which would not have happened if the sender waited long enough, and it results in retransmission due to a segment being delayed (but NOT lost) beyond TO. This produces serious end-to-end TCP performance degradation [Eom 02, Lud 00].

One of the main reasons for the delay spikes to occur in a wireless environment is congestion and the lack of mechanisms through which the sender can detect or be informed about these congestions, and consequently prohibits SRs. However, there are other reasons for the delay spikes to occur in a wireless environment, these may include [Fu 03, Gur 01b]:

1. The handoff of a mobile host between cells requires the base station to do channel allocation before data can be transmitted from the mobile host. This causes segments at the mobile host to be queued until the completion of the channel allocation, giving rise to sudden extra delay (in addition to the normal RTT).
2. The physical disconnection of the wireless link during a hard handoff will also result in a sudden increase of the RTT.
3. A Radio Link Control (RLC) layer between the Logical Link Control (LLC) and Multiple Access Control (MAC) layers, to carry out retransmission at the link layer (for error recovery) in wireless mobile networks (such as GPRS and CDMA2000), may result in delay spikes due to repeated retransmission attempts during link outages and periods of high link errors.
4. Higher-priority traffic, such as circuit-switched voice, can preempt (block) the data traffic temporarily. The duration of this blocking may be very long as compared to TCP's RTT estimate.

SFRs affect TCP performance in that the TCP sender unnecessarily reduces its load and unnecessarily retransmits a segment, i.e., performs a SR.

## 1.5 Modeling of TCP

There are three basic techniques for performance evaluation of TCP connection in wireless networks; these are [Kim 07, Voi 07, Mal 06, Mas 06, Hes 05, Lul 04, Kam 03, Ven 03, Bac 02]:

1. Experimental measurement
2. Computer simulation
3. Mathematical modeling of the TCP algorithms

Experimental measurement is not widely used to evaluate the performance of TCP connection, due to its difficulty, cost, limited flexibility, etc. On the other hand, due to the enormous development in computing resources and tools, computer simulation and mathematical modeling are extensively used to evaluate the performance of TCP connection algorithms. They have an equivalent importance as they are usually validating each other. In this work, we are concerned with analytical model of TCP connection, in particular TCP Reno.

### 1.5.1 Essential of TCP modeling

A number of notable models for TCP have been developed, which either shed light on a particular aspect of the protocol or add a new level of generality to the process of modeling transport control within the Internet. It is very useful, however, to consider the similarities of all these models before focusing attention on any one particular model, as this allows keeping the key features of the model in mind and not getting lost in the details of a specific model.

Earlier versions of the protocol do exist, which may not necessarily contain the features that are classed here as essential for proceeding with TCP modeling. Future TCP flavors may also depart from these essentials. In addition, there are some important dynamics of TCP, such as its Slow-Start procedure, and other phenomena, such as loss of ACK packets in queues, which are not generally included in mathematical models.



All TCP connections commence in slow-start and many spend their entire lives in Slow-Start, because only a few kilobytes of data are being transferred. Thus, it is important to understand that models do have their limitations in reflecting reality. However, the essential features that are listed here form the foundation of TCP as they know it and provide a starting point from which other models can be developed. There are two key processes that a model of TCP needs to include [Has 03]:

- i. The dynamics of the window that defines the number of packets a TCP source can convey into the network.
- ii. The Packet-Loss process that indicates current traffic loads or congestion within the network.

One thing to notice about these processes is that they are both observed from the reference point of the TCP sender. This is obvious for window size, which is controlled by an algorithm within the source itself. The packet-loss is also observed by the source. The loss process does not arise from any one particular node in the network but can be triggered by any node along the path of the TCP connection, with the source node observing the loss process as aggregation of information being generated along the connection path.

- i. Window dynamics

The typical symbol for the current window size is  $W(t)$ . The essential dynamics of this window size are its linear increase and multiplicative decrease [Has 03]. During the interval in which TCP receives information (i.e., packets are not being lost in the network), TCP increases its window linearly. When the source deduces that a packet has been lost, it reduces its window by a factor of the current window size (i.e., multiplicatively). Implementation of TCP normally increases the window by one packet each round trip (in the linear increase phase) and reduces the window size by half in the event of a packet-loss. Although, these parameters can be generalized in mathematical models of TCP, some models have been developed using the packet transmission (sending) rate  $S(t)$ , as this can ease the analysis that follows the development of the model.

The standard assumption in this case is that the window size is related to transmission rate by RTT:

$$S(t) = \frac{W(t)}{RTT} \quad (1.1)$$

Where  $S(t)$  is the sending rate in packets/sec,  $W(t)$  is the window size, and RTT is the round trip time. This does assume that increasing  $S(t)$  has negligible effect on queuing delays at nodes within the network, so that the RTT is effectively constant. Regardless of whether a model uses  $W(t)$  or  $S(t)$ , all models incorporate the Linear Increase and Multiplicative Decrease (LIMD) dynamics of TCP.

ii. Packet-Loss process

The other main component of a TCP model is a Packet-Loss process, which triggers the TCP source to reduce its window size. As previously mentioned, this process aggregates information regarding network conditions at all nodes along the path of the TCP connection. The particular TCP connection being considered is competing for network resources, along its path, with other TCP connections that have routes intersecting with this path. It is also competing for network bandwidth with other network traffic in general. These variations in traffic load introduce uncertainty into the arrival of Packet-Loss information at the TCP source.

This typically can be modeled as a stochastic process, either with regard to the probability  $p$  of losing a particular packet in the network or the intervals between instances when lost packets are detected. The key point is that models usually incorporate the arrival of Packet-Loss information, with the TCP source responding by decreasing its window. In fact, they do not necessarily need to consider the information being returned from the network as confirmation that packets have not been lost. Network information can take the form of explicit notification regarding congestion within the network, although individual congestion messages are most likely to still be coded as binary information.

Regardless of whether the information is packet-loss or explicit congestion information, TCP models must respond to the stream of network load information that is aggregated along the connection path.

## 1.6 Motivation

TCP was initially designed for wired networks, therefore, a number of notable mechanisms have been proposed in the literature to improve the performance of TCP in such networks. For most of these mechanisms, analytical models have developed to predict and investigate their performance in terms of the sending rate and throughput of TCP during different applications and wired networks environments. Unfortunately, these mechanisms demonstrate a poor performance in wireless networks due to the presence of Packet-Loss and Delay Spikes (Long Delay Cycles), which are especially more frequent in such networks.

Presence of Long Delay Cycles lead to STs, and consequently to SFRs, which produce serious end-to-end TCP performance degradation. However, since the emergence of wireless networks, new mechanisms have developed to enhance the performance of TCP in presence of STs and SFRs. Consequently, new and adequate analytical models need to be developed to accommodate these new TCP mechanisms. This is because none of the existed models for wired networks considers the effect of ST and SFR on the steady state sending rate and throughput of TCP. This may be due to the fact that:

- i. ST and SFR do not occur frequently in wired networks.
- ii. ST and SFR are considered to be a transient state in a wired network, and thus cannot produce much impact on the steady state performance of TCP.

Practically, in wireless networks, STs and SFRs are more frequent and must be explicitly modeled to accurately estimate the steady state sending rate, throughput and utilization efficiency of TCP.

## 1.7 Statement of the Problem

There are a number of mathematical models that have been developed throughout the years for evaluating the performance of a TCP connection in wireless networks, but they all have their own drawbacks and limitations, due to the approximation they assumed during the modeling process, e.g., effects of Long Delays, frequency of occurrence of Long Delays, Slow-Start stage, Acknowledgement (ACK) mechanism, etc.

The main objectives of this work can be summarized as follows:

1. Develop a new analytical model that can be used to evaluate the performance of TCP Reno in a realistic wireless environment that suffers from a wide-range of Packet-Loss (PL) and Long Delay Cycles (LDCs), namely, the PLLDC model. The new model is based on the stochastic model of TCP congestion control and avoidance that was developed by J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, therefore, it is referred to as PFTK model after the initials of the last names of the authors [Dun 06, Has 03, Pad 00].
2. Model the TCP Reno Sending rate ( $S$ ), Throughput ( $T$ ), and Utilization factor ( $U$ ) as a function of environment- and system-driven parameters. The former includes: Packet-Loss rate ( $p$ ), duration of the Long Delay ( $D$ ), Interval between Long Delays ( $I$ ), and Round Trip Time ( $RTT$ ), while the latter includes: Timeout ( $T_o$ ), Slow-Start Threshold at the end of a Long Delay ( $SST$ ), number of packets Acknowledged by one ACK packet ( $b$ ), and the receiver's maximum congestion Window size ( $W_m$ ).
3. Validate the results obtained by the PLLDC model against results obtained from the well-known PFTK model and the widely-used NS-2 network simulator.
4. Use the new analytical model to investigate the performance of TCP Reno under different network environments. The performance of the TCP Reno is mainly evaluated in terms of estimating the variation of  $S$ ,  $T$ , and  $U$  with  $p$  for various values of  $D$ ,  $I$ , and  $RTT$ .

## 1.8 Thesis Organization

This chapter provides an introduction to the general domain of this thesis which pertains to: TCP description, TCP sender analysis, TCP receiver analysis, TCP header format, congestion of TCP, TCP Reno, performance and modeling of TCP. The rest of this thesis is organized as follows:

Chapter 2 reviews some of the most recent work that is related to mathematical modeling of TCP in wireless networks.

Chapter 3 introduces features and modeling of the stochastic TCP congestion control model, namely, the PFTK model, which was proposed for predicting the steady-state  $S$  and  $T$  of TCP Reno, in a wireless environment that suffers from a wide range of Packet-Loss process. Then, this chapter presents the detailed description, features, and modeling of PLLDC model, and how it is used to derive expressions for estimating the performance of TCP Reno in a realistic wireless network environment suffering from a wide range of Packet-Loss (PL) and Long Delay Cycle (LDC). Also, in Chapter 3, an expression is derived for calculating the Utilization factor ( $U$ ), which is defined as the ratio between  $T$  and  $S$ .

Chapter 4 presents a description of three scenarios that are performed to validate the PLLDC model and evaluate the performance of TCP Reno in various network operation conditions. The results obtained from our PLLDC model are validated against results obtained from the PFTK model and the NS-2 network simulator. The results obtained for these three scenarios are presented in tables and graphs. Also, in this chapter, the results obtained are discussed.

Finally, in Chapter 5, based on the results obtained from the different simulations, conclusions are drawn, and suggestions and recommendations for future work are pointed-out.

# Chapter 2

## Literature Review

### 2.1 Introduction

TCP is currently the most widely used transport protocol in packet networks with primary responsibility for congestion control. In the absence of any explicit information about the network configuration, TCP achieves its congestion control objective by attempting to drive the network to the point of full utilization (by increasing the rate at which it releases packets to the network) while continuously monitoring the network for signs of congestion and lowering the rate if congestion is detected.

Packet loss is traditionally used as a signal of congestion; however, this information to the TCP sender is only implicitly provided by the network, and hence TCP employs schemes to infer packet loss; either by the expiry of a retransmission timer leading to a timeout or the reception of multiple duplicate Acknowledgments (ACKs) with the same expected packet number, termed duplicate ACK (DUPACK) detection.

The original intent behind TCP's congestion control law is to assume that the only cause of any inferred packet loss is congestion. Thus situations where the packet loss inference is non congestion related such as due to unreliable data links (as is characteristic for wireless channels), packet re-ordering or sudden large increase in propagation delays will needlessly trigger the congestion avoidance mechanisms in TCP and adversely affect the end-to-end throughput. Consequently there is a growing literature on modeling and performance evaluation of TCP performance where congestion loss is not dominant or, at least, not the only source of packet loss.

However, in this chapter, we provide a literature review on a number of mechanisms that have been proposed to improve the performance of TCP in the presence of a Spurious Timeout (STO) or a Spurious Fast Retransmission (SFR), and any other parameter that may degrade the TCP performance for different types of networks.

## 2.2 TCP Literature Review

**D. Malone et. al.** [Mal 08] investigated STOs in 802.11 infrastructure (access point) wireless networks. Though timeouts can be a problem for uploads from an 802.11 network, these timeouts are not spurious but are caused by a bottleneck at the access point. Once this bottleneck is removed, they found that STOs are rare, even in the face of large changes in numbers of active nodes.

They used experimental measurements to investigate the role of TCP timeouts in 802.11 infrastructure mode WLANs. They showed explicitly that the unfairness demonstrated in [Ng 05] between flows can be attributed to TCP timeouts, mainly due to TCP ACK losses at the Access Point (AP) buffer, caused by the MAC assigning too few transmission opportunities to TCP ACKs queued at the AP. They also demonstrated that once the AP has sufficient access to the medium to transmit TCP ACKs, TCP timeouts are rare.

**M. Abdelhafez et. al.** [Abd 07] studied the behavior of TCP based worms in mobile ad hoc networks (MANETs). They developed analytical models for the spread of TCP worms in the MANET environment that account for payload-size, bandwidth sharing, radio range, nodal density, packet discards and several other parameters specific to MANETs. They presented numerical solutions for the models and verified the results using high fidelity packet-level simulations. The results showed that the analytical model developed matches the results of the packet-level simulation in all cases except when topologies result in a high probability of disconnected clusters. Their simulation studies showed that under many cases, due to the resource constrained nature of the MANET and its underlying wireless layers, the TCP-based worms rapidly become self-throttling, which may benefit the design of effective mitigation technologies in such critical networking environments.

**R. Dunaytsev, Y. Koucheryavy and J. Harju** [Dun 06] presented an analytical model of TCP Reno throughput as a function of loss event rate, average round trip time (RTT), average retransmission timeout value, and receiver window size based on the PFTK model [Pad 00]. In their model, they refined the PFTK

model by careful examination of fast retransmit/fast recovery dynamics in the presence of correlated losses and taking into consideration slow start phase after timeout.

The accuracy of the proposed model was validated against simulation results and compared with those of the PFTK-model. Simulation results show that the model gives a more accurate estimation of TCP Reno throughput in the presence of correlated losses than the PFTK-model.

**A. Argyriou and V. Madisetti** [Arg 06a, Arg 06b] presented a joint performance evaluation model of TCP and TCP-friendly rate control (TFRC) protocol [Han 03], with the underlying IP-based mobility protocols. They developed stochastic models that can characterize the protocol performance during handoffs between heterogeneous wireless networks like WLAN, cellular, or WMAN. They presented performance evaluation results for validating the developed models under a set of different handoff scenarios. Their developed model can be utilized as a basis for further analytical evaluation of new mobility management protocols, allowing thus a fast and accurate comparison.

The model was found to be accurate for TCP in both the cases where Hierarchical Mobile IP (HMIP) and Mobile IP with Route Optimization (MIP-RO) were used as the underlying mobility management protocols. However, the TFRC model predicts the expected throughput with even better accuracy, due to the simpler protocol algorithms. For example the worst case error for the TCP model was nearly 22% while for the TFRC model it was 13%. They also introduced the notion of the “recovery period”. The slow-responsive rate control algorithm of TFRC, requires less time in order to recover when compared with TCP. However, they found that as the disruption time is increased, TFRC suffers from more packet losses than TCP, due to the slow-responsive algorithm.

**E. Altman and et. al.** [Alt 05] presented a mathematical analysis of the Multiplicative Increase Multiplicative Decrease (MIMD) congestion control algorithm in the presence of random losses. Random losses are typical to



wireless networks but can also be used to model losses in wire line networks with a high bandwidth-delay product. The Laplace–Stieltjes transform of the equivalent queue is then shown to directly provide the throughput of the congestion control algorithm and the higher moments of the window size.

For window independent losses, an exact expression can be obtained for the steady state probability distribution of the window size, and the throughput of the connection. For window dependent losses, an approximate expression, analogous to the square root formula for standard TCP, can be used to compute the throughput. This approximation is observed to be close to the actual throughput obtained from simulations. They validate their finding on scalable TCP using ns-2 simulations.

**A. Kesselman and Y. Mansour** [Kes 05a] showed that the optimal Retransmission Timeout (RTO) that maximizes the TCP throughput need to depend on both RTT and TCP window size. Intuitively, the larger the TCP window size, the longer the optimal RTO. They derived the optimal RTO for several RTT distributions. They also demonstrated that an important advantage of their algorithm is that it can be easily implemented based on the existing TCP timeout mechanism. However, before their work, i.e., in the previous TCP implementations, RTO is a function of the RTT alone.

**A. Kesselman and Y. Mansour** [Kes 05b] proposed a novel congestion control algorithm that achieves high bandwidth utilization providing fairness among competing connections and, on the other hand, is sufficiently responsive to changes of available bandwidth. The main idea of the algorithm is to use adaptive setting for the additive increase/multiplicative decrease (AIMD) congestion control scheme, where parameters may change dynamically, with respect to the current network conditions.

The proposed algorithm attains almost optimal utilization in a steady state providing fairness between competing connections and at the same time responds quickly on bandwidth changes. Even though the algorithm cannot be

implemented without additional work, they found that the proposed scheme may suggest new directions for further improvement of the current TCP congestion control.

**D. Leith and P. Clifford** [Lei 05] investigated the use of the 802.11e MAC EDCF to address transport layer unfairness in WLANs. A simple solution was developed that uses the 801.11e AIFS and  $CW_{min}$  parameters to ensure fairness between competing TCP uploads. An analytic model of TCP transport over the modified channel was developed in order to study the fairness properties of the proposed scheme. In addition to fairness between competing TCP flows, consideration was extended to include characteristics of TCP flows such as RTT unfairness and responsiveness and they observed that TCP flows with a wireless bottleneck link exhibit quite different properties from flows with a wired bottleneck.

**R. Ludwig and A. Gurtov** [Lud 05] based on the Eifel detection algorithm developed, a response algorithm for TCP, namely, the Eifel response algorithm to provide a way for a TCP sender to respond to a detected STO. It adapts the retransmission timer to avoid further STOs and (depending on the detection algorithm) can avoid the often unnecessary go-back-N retransmits that would otherwise be sent. In addition, the Eifel response algorithm restores the congestion control state in such a way that packet bursts are avoided.

**A. Ng, D. Malone, and D. Leith** [Ng 05] presented measurements made using an 802.11e wireless test-bed. They demonstrated experimentally how the new 802.11e QoS parameters behave in their test-bed. They described the testing methodology used to validate the operation of the 802.11e TXOP, AIFS and  $CW_{min}$  parameters and compared the experimental results to existing analytical models. They also discussed a number of practical issues encountered during the measurements. They then used the test-bed to demonstrate some known problems with TCP's performance caused by cross-layer interaction between the TCP congestion control algorithm and the MAC layer CSMA/CA contention mechanism. Finally, they studied how these problems can be mitigated using the flexibility provided by the 802.11e parameters via the scheme suggested in [Lei 05].

**E. Blanton and M. Allman** [Bal 04] developed conservative methods of using information from the TCP and Stream Control Transmission Protocol (SCTP) to identify unnecessary retransmissions for various applications. The TCP and SCTP provide information of duplicate segment receipt through Duplicate Selective Acknowledgement (DSACKs) and duplicate Transmission Sequence Number (TSN) information, respectively.

**A. Budhiraja et. al.** [Bud 04] described a new analytic model for the dynamic behavior of TCP windows as they respond to congestion indicated by data loss in the network. They developed a stochastic differential equation to describe the dynamic evolution of the congestion window size of a single TCP session over a network.

The model takes into account recovery from packet losses with both fast recovery and time-outs, boundary behavior at zero and maximum window size, and slow-start after time-outs. It is based on stochastic differential equations and allows computing throughput as well as the complete probability distribution function (PDF) for effective window sizes. They solved the differential equation to derive the distribution of the window size in steady state. The results computed from this model had been compared with NS simulations of TCP behavior in a realistic network environment and were found to produce comparable results.

**A. Abouzeid and S. Roy** [Abo 03] presented an analytical model of TCP over an end-to-end path with random abrupt RTT changes. The RTT variations are modeled as a stochastic process (semi-Markov process); this allows analytical estimation of steady-state TCP throughput based on renewal reward theory [16]. They analytically quantify and compare between the degree of degradation of the steady-state average throughput and window size due to spurious retransmissions for the different versions of TCP (Reno/NewReno versus Tahoe). The modeling methodology is used for evaluating different design alternatives for TCP for highly dynamic networks.

Their findings can be summarized as follows:

- i. The analytical model matched simulation results for a wide range of parameters with a maximum error of less than 20% (mean error of less than 10%).
- ii. OldTahoe outperformed Reno in a considerable number of cases, yielding an improvement in the throughput of up to 100%.
- iii. Setting the DUPACK threshold to half the maximum path bandwidth delay product improves Reno's performance to match that of OldTahoe,
- iv. The degradation of TCP throughput performance as compared to ideal TCP becomes more evident at higher advertised windows.
- v. The rate at which the RTT changes has a drastic effect on TCP's throughput.
- vi. The analytical framework provides a model for the evaluation of different window-based control protocols that may be more adaptive to the variations of the RTT process.

**H. T. Kunga, K. S. Tanb, and P. H. Hsiao** [Kug 03] described a congestion control method for TCP that adjusts the transmission rate of a TCP connection by changing not only the congestion window size as in normal TCP, but also by delaying the transmission of packets at the sender. This method is similar to TCP with Sender-based Delay Control (SDC). SDC can keep the window size of a TCP connection above a certain threshold even when its fair share of bandwidth is arbitrarily small. Since TCP fast retransmit and recovery is likely to work when the window size of the connection is sufficiently large, the new method can result in reduced frequency of TCP timeouts for the connection. In particular, SDC allows many TCP flows to share a link without experiencing many timeouts. In addition, SDC can reduce a well-known TCP bias against connections with large RTTs.

**S. Floyd** [Flo 03], in RFC 3649, proposed a modification of TCP congestion control that adapts the increase strategy and make it more aggressive for high bandwidth links (i.e., for large window size).

**S. Fu and M. Atiquzzaman** [Fu 03] proposed an analytical model of TCP Reno sending rate and throughput as a function of packet error rate and characteristics of STOs. The accuracy of the proposed model was validated against simulation results. The effectiveness of the model was compared with those of previous models, and has been found to be more accurate than PFTK model in estimating the steady state sending rate and throughput of TCP in presence of frequent long delays.

The main contributions of Fu and Atiquzzaman can be summarized as follows:

- Developed an analytical model of TCP performance by explicitly considering STO effect.
- Compared the effectiveness of the proposed analytical model with that of PFTK model, and found that the proposed model is much more accurate for estimating TCP performance in the presence of frequent long delays.
- The model has been validated against simulation results.

The model proposed by Fu and Atiquzzaman was expected to significantly contribute to further studies as follows:

- i. There is always a fundamental trade off between the rapidness of detection of true losses versus the risk of unnecessary retransmissions when designing a RTO calculation algorithm or setting related parameters. For example, the TCP parameter  $RTO_{min}$ , the lower bound of the RTO value, has a significant impact on the effectiveness of the RTO estimator. There is no existing method to optimally set  $RTO_{min}$ , and the current practice is to set it to twice the clock granularity. Since our proposed model considers the effect of STO, it can assist in determining an appropriate value of  $RTO_{min}$ .

- ii. There is an increasing research interest to study the interaction between TCP and lower layer protocols in wireless environments. The settings of lower layer protocols, such as handoff schemes in Mobile IP and retransmission schemes at the link layer, have a non-trivial impact on the frequency of TCP STOs. The proposed model can facilitate the fine-tuning of these settings in a more coordinated fashion in order to achieve an optimal performance.
- iii. The modifications to TCP were made to alleviate the effects of STO, the proposed model provided a framework for evaluating the impact of the modifications, and to compare the performance of the modified TCP with previous versions of TCP.

**N. Moller and K. Henrik** [Mol 03] provided the fundamental assumption of the TCP protocol is that packet losses indicate congestion on the network. This is a problem when using TCP over wireless links, because a noisy radio transmission may erroneously indicate congestion and thereby reduce the TCP sending rate. They modeled two partial solutions, namely, the power control and link-level retransmissions, to improve the quality of the radio link. By modeling these two lower layers of control loops, they derived an analytical model of the delay distribution for IP packets traversing a link. They investigated the effect on TCP, in particular the performance degradation due to STOs and SFRs caused by delays and reorder on the link. The models allowed the quantification of the throughput degradation. The results indicated that tuning of link-level control or TCP interacts, or both of them, was needed in order to improve performance.

**A. Gurtov** [Gur 01a] discussed optimizations for a TCP sender that are most helpful in the presence of delays spikes, but are seemingly suitable for general deployment. The motivation for their work is increasing popularity of links (e.g. provided by cellular networks) that have delay spikes exceeding the usual link latency by several times. The effect of a delay spike on TCP Tahoe, Reno, NewReno and SACK is described. He recommended timing every segment and restarting the retransmit timer to achieve a more conservative RTO estimate

. Furthermore, it discussed how a series of DUPACKs should be treated.

**A. Gurtov** [Gur 01b] presented several contributions. First, he reported that long sudden delays during data transfers are not uncommon in the GPRS wireless WAN. Long sudden delays can lead to STOs and unnecessary retransmissions. Second, he showed that the NewReno algorithm increases the penalty of STOs and that an aggressive TCP retransmission timer may trigger a chain of spurious retransmissions. Third, he tested how four widely deployed TCP implementations recover from a STO and noticed that two of them had severe problems to recover. Finally, he discussed several existing ways to alleviate the problems.

**K. Li and et. al.** [Li 01] built a hybrid state-space-based model of TCP using differential equations and event-driven switches to study the effect on bandwidth sharing of interactions among a set of competing TCP-friendly flows. They modified the TCP model, using TCP's Additive-Increase Multiplicative Decrease (AIMD) congestion avoidance algorithm with different increase and decrease parameters, to create theoretically TCP-friendly protocols with various short term transmission rates. They proved that TCP-friendly flows result in a stable attractor if the backing off of flow transmission rates is synchronized. They performed a number of experiments using their model and using ns-2 simulator with unsynchronized backing off which showed unfairness among competing flows with different short-term behaviors.

**J. Widmer, R. Denda, and M. Mauve** [Wid 01] presented a survey of TCP-friendly congestion control schemes to summarize the state-of-the-art in this field of research and motivate further research on TCP friendliness. They defined TCP friendliness and outlined the design space for TCP-friendly congestion control. Existing single-rate protocols were discussed, and a detailed survey of multirate protocols was given. The article contained an evaluation of the strengths and weaknesses of the mechanisms presented. They also pointed to open problems and issues for future research and gave some concluding remarks.

**R. Ludwig and R. H. Katz** [Lud 00] proposed an enhancement to TCP's error recovery scheme, which they called the Eifel algorithm. It eliminates the retransmission ambiguity, thereby solving the problems caused by STOs and SFRs. It can be incrementally deployed as it is backwards compatible and does not change TCP's congestion control semantics. In environments where spurious retransmissions occur frequently, the algorithm can improve the end-to-end throughput by several tens of percent. The Eifel algorithm finally makes TCP truly wireless-capable without the need for proxies between the end points. Another key novelty was that the Eifel algorithm provided for the implementation of a more optimistic retransmission timer because it reduces the penalty of a STO to a single (in the common case) spurious retransmission.

**P. Kuusela et. al.** [Kuu 00] analyzed the dynamic behavior of a single RED controlled queue interacting with a large population of idealized TCP sources, i.e., sources obeying the rules of linear increase and multiplicative decrease. The aggregate traffic from this population is modeled in terms of the time dependent expected value of the packet arrival rate which reacts to the packet loss taking place in the queue. The queue is described in terms of the time dependent expected values of the instantaneous queue length and of the exponentially averaged queue length, for which they also derive a pair of differential equations. This provided which a complete model for the dynamics of the system which they used to explore transient and equilibrium behavior; the accuracy of the model is verified by comparison with simulated results.



## Chapter 3

# The Packet-Loss and Long Delay Cycles (PLLDC) Model

Chapter 2 demonstrates that during the last two decades, several researches have reported analytical models to predict the performance of TCP in wireless networks. But, all of them have their own drawbacks and limitations, mainly due to the approximation they assumed during the modeling process. Most of these models approximate or neglect the effects of some or all of the following parameters: the Packet-Loss rate, Long Delays, frequency of occurrence of Long delays, Slow Start stage, spurious fast retransmission caused by acknowledgement behavior and spurious timeouts, fast recovery, etc [Ho 08, Kim 07, Dun 06, Mas 06, Ven 03].

In this chapter, the researcher aims to develop a comprehensive analytical model that can be used to predict and evaluate the performance of TCP Reno in realistic wireless networks suffering from Packet-Loss (PL) and Long Delay Cycle (LDC), therefore, it is referred to it as the PLLDC model.

The proposed PLLDC model utilizes the stochastic TCP congestion control model, namely, the PFTK model, which was proposed by Padhye et. al. [Hou 08, Dun 06, Xin 06, Fu 03, Pad 00], to derive expressions for predicting the steady state sending rate ( $S$ ) and throughput ( $T$ ) of TCP Reno in presence of PL and LDC. Also, in this model, an expression is derived for calculating the utilization factor ( $U$ ), which is defined as the ratio between  $T$  and  $S$ . These expressions for  $S$ ,  $T$ , and  $U$  are derived as a function of eight input parameters.

The eight input parameters are categorized into two groups, each of four parameters, as follows:

1. Environment or network driven parameters, these include:
  - a. Packet-error rate ( $p$ ).
  - b. Duration of the Long Delay ( $D$ ).

- c. Interval between Long Delays ( $I$ ).
  - d. Round Trip Time (RTT) when there is no Long Delay ( $RTT$ ).
2. System or technology driven parameters, these include:
- a. Retransmission timeout or Timeout ( $T_o$ ).
  - b. Slow Start Threshold at the end of a Long Delay ( $SST$ ).
  - c. Number of packets acknowledged by one ACK packet ( $b$ ).
  - d. Receiver's maximum advertised congestion Window size ( $W_m$ ).

The rest of this chapter is organized as follows. In Section 3.1, a detail description of the PFTK model will be provided. The PFTK model characterizes both the fast retransmit and the timeout behavior of TCP Reno, and can accurately predict TCP performance over a wide range of Packet-Loss rates. The PLLDC modeling assumptions are summarized in Section 3.2. Section 3.3 introduces the basics of the PLLDC model. The modeling of the TCP Sending rate, Throughput, and Utilization factor is presented in Sections 3.4, 3.5, and 3.6, respectively. Finally, in this chapter, the implementation of the new model and its main input and computed parameters are defined.

## 3.1 The PFTK Congestion Control Model

### 3.1.1 Features of the PFTK model

This section presents a description of the stochastic model of TCP congestion control and avoidance that was developed by J. Padhye, V. Firoiu, D. Towsley, and J. Kurose; therefore, it was referred to it as PFTK model after the initials of the last names of the authors [Dun 06, Has 03, and Pad 00].

The main features of the PFTK model include:

1. It is a relatively simple analytical expression for predicting the sending rate of a saturated TCP sender, i.e., a steady-state flow with an unlimited amount of data to send.
2. It illustrates the congestion avoidance behavior of TCP and its impact on sending rate, taking into account the dependence of congestion avoidance on:
  - i. Acknowledgment (ACK) behavior.
  - ii. The manner in which Packet-Loss is inferred, whether by Duplicate Acknowledgments (DUPACK) detection and fast retransmits or by Timeout (TO).
  - iii. Limited receiver window size.
  - iv. Average Round Trip Time (*RTT*).
2. The model is based on the TCP Reno flavor as it is one of the more popular implementations in the Internet today.
3. In this model the congestion control Window size,  $W$ , is increased by  $1/W$  each time an ACK is received. Conversely, the window is decreased whenever a lost packet is detected, with the amount of the decrease depending on whether Packet-Loss is detected by DUPACK or by TO.
4. The model represents the congestion avoidance behavior of TCP in terms of rounds. A round starts with transmission of  $W$  packets, where  $W$  is the current size of the TCP congestion window. Once all packets falling within the congestion window have been sent, no other packets are sent until the first ACK is received for one of these  $W$  packets. This ACK reception marks the end of the current round and the beginning of the next round.

5. The duration of a round is equal to  $RTT$  and is assumed to be independent of the window size.
6. The time needed to send all the packets in a window is smaller than the  $RTT$ .
7. A Packet-Loss in a round is entirely independent of Packet-Loss in other rounds.
8. Packet-Losses are correlated among the back-to-back transmissions within a round: if a packet is lost, all remaining packets transmitted until the end of that round are also lost.

### 3.1.2 Modeling of the TCP Reno sending rate ( $S$ ) (PFTK model)

Many TCP receiver implementations send one cumulative ACK for consecutive packets (i.e., delayed ACK). Let  $b$  be the number of packets that are delayed and acknowledged by a single ACK. So that  $b=2$  if one ACK is sent for two consecutive packets received at the receiver. If  $W$  packets are sent in the first round and are all received and acknowledged correctly, then  $W/b$  ACKs will be received. Since each ACK increases the window size by  $1/W$ , the window size at the beginning of the second round is then  $W'=W+1/b$ . That is, during congestion avoidance and in the absence of loss, the window size increases linearly in time, with a slope of  $1/b$  packets per RTT.

In the following, the behavior of TCP in the presence of packet-loss is modeled. A Packet-Loss can be detected at the TCP sender in one of two ways [Has 03, Kam 03]:

- i. Reception of Triple-DUPACK (TD), which is denoted as a TD loss indication.
- ii. Timeout (TO), which is denoted as a TO loss indication.

The PFTK stochastic model of TCP congestion control was developed in three steps, corresponding to its operating regimes:

- i. Loss indications are exclusively triple-DUPACK (TD).
- ii. Loss indications are both triple-DUPACK (TD) and timeout (TO).
- iii. The congestion window size is limited by the receiver's advertised window.

For each of the above operating regimes, an expression was derived for estimating the TCP sending rate. A brief description and summary of the expressions are given below, while a detail description and derivation can be found in [Ho 08, Xin 06, Bac 02, Pad 00, Alt 00].

- i. Loss indications are exclusively triple-DUPACK (TD)

At this stage, the losses indications are assumed to be exclusively of type TD, and that the window size is not limited by the receiver's advertised flow control window. A sample path of the evolution of congestion window size is given in Figure (3.1). Between two TD loss indications, the sender is in congestion avoidance, and the window increases by  $1/b$  packets per round, and immediately after the loss indication occurs, the window size is reduced by a factor of two.

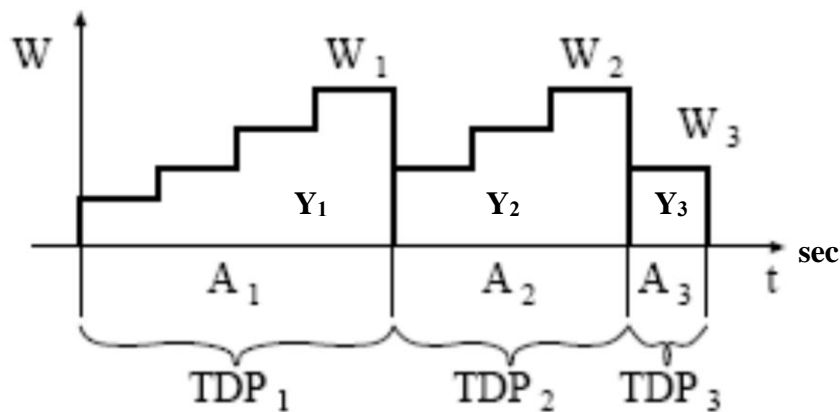


Figure (3.1). Evolution of window size over time when loss indications are TD .

The period between two TD loss indications is referred to it as a TD period (TDP) and denoted by ( $D_{TDP}$ ) as shown in Figure (3.1). A TDP starts

immediately after a TD loss indication as shown in Figure (3.2). For the  $i^{th}$  TDP, define  $S_{TDP}$  to be the number of packets sent in the period  $D_{TDP}$ , which is the duration of the period,  $W_i$  the window size at the end of the period, and  $p$  the probability that a packet is lost, given that either it is the first packet in its round or the preceding packet in its round is not lost. Due to the stochastic nature of the loss process, the long-term steady-state TCP sending rate as a function of  $p$  presented in average form,  $E[S]$ , can be expressed as:

$$E[S] = \frac{E[S_{TDP}]}{E[D_{TDP}]} \quad (3.1)$$

Where  $E[S_{TDP}]$  is the average number of packets sent during a TDP, and  $E[D_{TDP}]$  is the average duration of a TDP.

It is clear from the above equation that in order to calculate  $S$ , it is important, first, to derive an expression for calculating the average values of  $S_{TDP}$  and  $D_{TDP}$ . Mathematical expressions were derived by Padyhe et. el. for calculating both  $S_{TDP}$  and  $D_{TDP}$ , these are [Hes 05, Has 03, Yan 03, Pad 00]:

$$E[S_{TDP}] = \frac{1-p}{p} + \frac{2+b}{3b} + \sqrt{\frac{8(1-p)}{3bp} + \left(\frac{2+b}{3b}\right)^2} \quad (3.2)$$

$$E[D_{TDP}] = RTT \left( \frac{2+b}{6} + \sqrt{\frac{2b(1-p)}{3p} + \left(\frac{2+b}{6}\right)^2} + 1 \right) \quad (3.3)$$

Thus, the long-term steady-state sending rate of a TCP source when loss indications are exclusively TD is expressed as:

$$E[S] = \frac{\frac{1-p}{p} + \frac{2+b}{3b} + \sqrt{\frac{8(1-p)}{3bp} + \left(\frac{2+b}{3b}\right)^2}}{RTT \left( \frac{2+b}{6} + \sqrt{\frac{2b(1-p)}{3p} + \left(\frac{2+b}{6}\right)^2} + 1 \right)} \quad (3.4)$$

For small packet-loss rate, the above equation can be simplified to:

$$E[S] = \frac{1}{RTT} \sqrt{\frac{3}{2bp}} + o(1/\sqrt{p}) \quad (3.5)$$

Furthermore, for  $b=1$ , Eqn. (3.5) can be reduced to the equation of the inverse square-root  $p$  law or periodic model, which is an important and well-known relationship governing the performance of TCP. In particular it shows that the transmission rate of a TCP source is inversely related to RTT and the square root of  $p$  [Has 03].

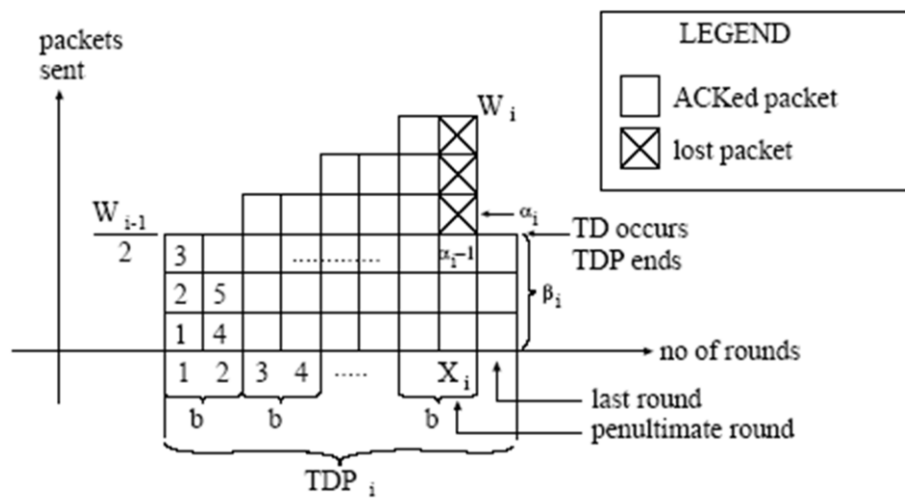


Figure (3.2). Packets sent during a TDP.

ii. Loss indications are triple-DUPACK (TD) and timeout (TO)

In practice, it has been realized that in many cases the majority of window decreases are due to TO, rather than fast retransmit. Therefore, a good mathematical model should capture TO loss indications. This occurs when packets (or ACKs) are lost, and less than TD are received. In normal operation, the sender waits for a period of time (TO period), which is denoted by  $T_o$ , and then retransmits non acknowledged packets. Following a TO, the congestion window is reduced to one, and one packet is thus resent in the first round after a TO. In the case that another TO occurs before successfully retransmitting the

packets lost during the first TO, the period of TO doubles to  $2T_0$ ; this doubling is repeated for each unsuccessful retransmission until a TO period of  $64T_0$  is reached, after which the TO period remains constant at  $64T_0$ .

Figure (3.3) illustrates an example on the evolution of congestion window size. In this case, due to the stochastic nature of the packet-loss process,  $S$ , can be calculated by:

$$E[S] = \frac{E[S_T]}{E[D_T]} = \frac{E[S_{TD} + S_{TO}]}{E[D_{TD} + D_{TO}]} \quad (3.6)$$

Where

$D_{TO}$  the duration of a sequence of TOs,

$D_{TD}$  the time interval between two consecutive TO sequences,

$D_T$  the sum of  $D_{TO}$  and  $D_{TD}$  (i.e.,  $D_T = D_{TD} + D_{TO}$ ),

$S_{TO}$  the number of packets sent during  $D_{TO}$ ,

$S_{TD}$  the number of packets sent during  $D_{TD}$ ,

$S_T$  the number of packets sent during  $D_T$  (i.e.,  $S_T = S_{TD} + S_{TO}$ ).

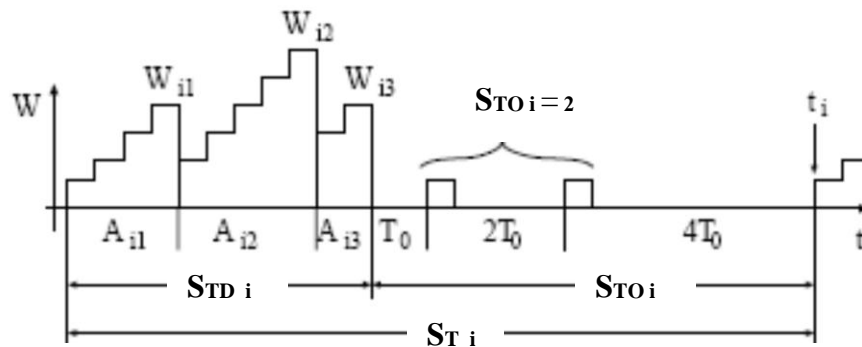


Figure (3.3): Evolution of window size when loss indications are TD and TO.

Another form of Eqn. (3.6) was derived in [Pad 00], which is expressed as:



$$E[S] = \frac{E[S_{TDP}] + Q * E[R_{NP}]}{E[D_{TDP}] + Q * E[D_{TO}]} \quad (3.7)$$

In Eqn. (3.7),  $E[S_{TDP}]$  and  $E[D_{TDP}]$  are as given in Eqns. (3.2) and (3.3), respectively.

$$Q \approx \hat{Q}(E[W]) = \min \left( 1, \frac{3}{E[W]} \right) \quad (3.8)$$

and 
$$E[W] = \frac{2+b}{3b} + \sqrt{\frac{8(1-p)}{3bp} + \left(\frac{2+b}{3b}\right)^2} \quad (3.9)$$

$E[R_{NP}]$  denotes the number of retransmitted packets during the TO period in one normal period ( $NP$ ), and it is computed by:

$$E[R_{NP}] = \frac{1}{1-p} \quad (3.10)$$

$E[D_{TO}]$  denotes the average duration of a TOs sequence excluding retransmissions, which can be computed by:

$$E[D_{TO}] = T_o \frac{f(p)}{1-p} \quad (3.11)$$

Where  $T_o$  is the TO and  $f(p)$  is given by:

$$f(p) = 1 + p + 2p^2 + 4p^3 + 8p^4 + 16p^5 + 32p^6 = 1 + \sum_{i=1}^6 2^{i-1} p^i \quad (3.12)$$

Substituting Eqns. (3.2), (3.3), (3.10), and (3.11) into Eqn. (3.7) yields the following expression for calculating S:

$$E[S] = \frac{\frac{1-p}{p} + \frac{2+b}{3b} + \sqrt{\frac{8(1-p)}{3bp} + \left(\frac{2+b}{3b}\right)^2} + Q \cdot \frac{1}{1-p}}{RTT \left( \frac{2+b}{6} + \sqrt{\frac{2b(1-p)}{3p} + \left(\frac{2+b}{6}\right)^2} + 1 \right) + Q \cdot T_o \cdot \frac{f(p)}{1-p}} \quad (3.13)$$

Where  $Q$  and  $f(p)$  are calculated by using Eqns. (3.8) and (3.12), respectively.

iii. Impact of window limitation

So far, no limitation is considered on the congestion window size. In practice, however, at the beginning of TCP flow establishment, the receiver advertises a maximum buffer size which determines a maximum congestion window size,  $W_m$ . As a consequence, during a period without loss indications, the window size can grow up to  $W_m$ , but will not grow further beyond this value. An example

of the evolution of window size is depicted in Figure (3.4). To simplify the analysis of the PFTK model, the following assumption is assumed. Let  $W_u$  denote the unconstrained window size, the mean of which is derived by Padhye et. al. [Pad 00] and it is given by Eqn. (3.9) with  $W_u$  replaces  $W$ .

It is assumed that if  $E[W_u] < W_m$ , then the approximation  $E[W] \approx E[W_u]$  is satisfactory. In other words, if  $E[W_u] < W_m$ , the receiver window limitation has negligible effect on the long-term average of the TCP sending rate, and thus the sending rate is given by Eqn. (3.13).

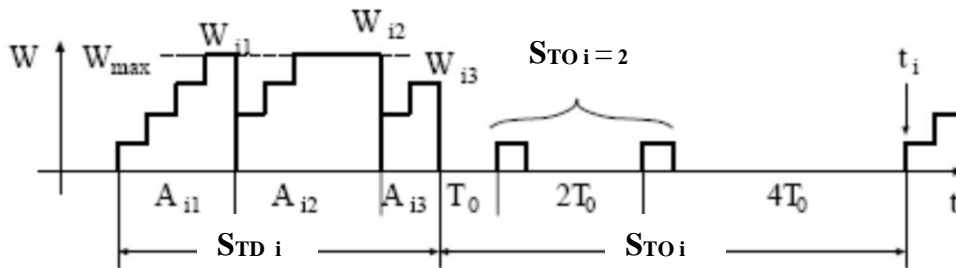


Figure (3.4). Evolution of window size limited by  $W_m$ .

On the other hand, if  $W_m \leq E[W_u]$ , the approximation  $E[W] \approx W_m$  can be considered as a satisfactory approximation. In this case, consider an interval  $D_{TD}$  between two time-out sequences consisting of a series of TDPs as in Figure (3.5). During the first TDP, the window grows linearly up to  $W_m$  for  $H$  rounds, afterwards remains constant for  $L$  rounds, and then a TD indication occurs. The window then drops to  $W_m/2$ , and the process repeats. According to the above discussion Padhye et. al. derived an expression for  $S$ , which is expressed as:

$$E[S] = \frac{\frac{1-p}{p} + W_m + Q_m \cdot \frac{1}{1-p}}{RTT \left( \frac{b}{8} W_m + \frac{1-p}{p W_m} + 2 \right) + Q_m \cdot T_o \frac{f(p)}{1-p}} \quad (3.14)$$

Where all variables are as defined above, except  $Q_m$ , which is different from  $Q$  and it is given as a function of  $W_m$  and it is expressed as:

$$Q_m \approx \hat{Q}(w) = \min \left( 1, \frac{(1-(1-p)^3(1+(1-p)^3(1-(1-p)^{w-3})))}{1-(1-p)^w} \right) \quad (3.15)$$

In conclusion, the complete characterization of TCP sending rate,  $S$ , can be calculated according to the following general expression:

$$E[S] = \begin{cases} \frac{\frac{1-p}{p} + \frac{2+b}{3b} + \sqrt{\frac{8(1-p)}{3bp} + \left(\frac{2+b}{3b}\right)^2} + Q \cdot \frac{1}{1-p}}{RTT \left( \frac{2+b}{6} + \sqrt{\frac{2b(1-p)}{3p} + \left(\frac{2+b}{6}\right)^2} + 1 \right) + Q \cdot T_o \frac{f(p)}{1-p}} & \text{for } E[W_u] < W_m \\ \frac{\frac{1-p}{p} + W_m + Q_m \cdot \frac{1}{1-p}}{RTT \left( \frac{b}{8} W_m + \frac{1-p}{p W_m} + 2 \right) + Q_m \cdot T_o \frac{f(p)}{1-p}} & \text{Otherwise} \end{cases} \quad (3.16)$$

Where  $Q$ ,  $f(p)$ ,  $E[W_u]$ , and  $Q_m$  can be calculated by Eqns. (3.8), (3.12), (3.9), and (3.15), respectively. Eqn. (3.16) is referred to as the ‘‘PFTK full model’’, which can be approximated to:

$$E[S] = \min \left( \frac{W_m}{RTT}, \frac{1}{RTT \sqrt{\frac{2bp}{3} + T_o \min \left( 1, 3\sqrt{\frac{3bp}{8}} \right)} p(1+32p^2)} \right) \quad (3.17)$$

Eqn. (3.17) is referred to as the ‘‘PFTK approximate model’’.

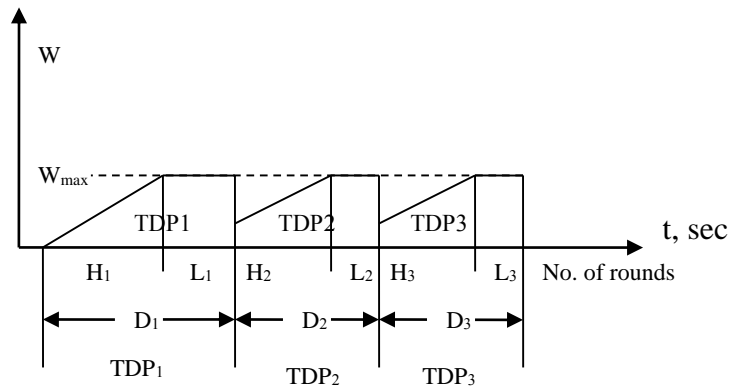


Figure (3.5). Fast retransmit with window limitation.

### 3.1.3 Modeling of the TCP Reno throughput ( $T$ ) (PFTK model)

In the previous sections, we have focused our attention on the derivation of a formula for estimating the sending rate of TCP Reno. The steady-state performance of TCP Reno may also be characterized by throughput ( $T$ ), which is the amount of data received by the receiver in unit time. The same analysis that has been used to derive Eqn. (3.16) to calculate  $S$ , can be easily modified to calculate throughput. It can be seen that to calculate  $T$ , instead of  $S$ , it only needs to modify the numerator of Eqn. (3.7). So that, the number of packets that makes it to the receiver in a TDP ( $E[S'_{TDP}]$ ) and the number of packets sent in the time-out sequence ( $E[R'_{NP}]$ ) are needed to be calculated. The throughput, then, can be calculated as:

$$E[T] = \frac{E[S'_{TDP}] + Q \cdot E[R'_{NP}]}{E[D_{TDP}] + Q \cdot E[D_{TO}]} \quad (3.18)$$

Since only one packet makes it to the receiver in a time-out sequence (i.e., the packet that ends the time-out sequence), it is evident that

$$E[R'_{NP}] = 1 \quad (3.19)$$

To calculate the number of packets that reaches the receiver in a TDP, consider Figure (3.2). The TD event is induced by the loss of packet. Assume the window size be  $W$ , when the loss occurs. Then, the number of packets received by the receiver is

$$E[S_{TDP}^i] = \frac{1-p}{p} + \frac{E[W]}{2} \quad (3.20)$$

From Eqns. (3.19) and (3.20) along with the analysis for  $E[W]$  and  $Q$  from Section (3.1.1),  $T$  can be expressed as:

$$E[T] = \begin{cases} \frac{\frac{1-p}{p} + \frac{E[W]}{2} + Q(p, E[W])}{RTT(E[W]+1) + Q(p, E[W]) \cdot T_o \frac{f(p)}{1-p}} & \text{for } E[W_u] < W_m \\ \frac{\frac{1-p}{p} + \frac{W_m}{2} + Q(p, W_m)}{RTT\left(\frac{W_m}{4} + \frac{1-p}{pW_m} + 2\right) + Q(p, W_m) \cdot T_o \frac{f(p)}{1-p}} & \text{Otherwise} \end{cases} \quad (3.21)$$

Where  $E[W]$  and  $f(p)$  are defined in Eqns. (3.9) and (3.12), respectively. However,  $Q(p, w)$  was derived in [Pad 00] as:

$$Q(p, w) = \min\left(1, \frac{(1-(1-p)^3(1+(1-p)^3(1-(1-p)^{w-3})))}{1-(1-p)^3}\right) \quad (3.22)$$

### 3.2 The PLLDC Modeling Assumptions

The assumptions that have been made for developing the new PLLDC analytical model of TCP with packet-loss and long delay cycles can be summarized as follows [Kim 07, Mas 06, Xin 06, Chr 05, Lul 04, Fu 03, Lud 01, Bac 02, Pad 00]:

- To isolate only the impact of packet-losses and long delays on TCP, it is assumed that the sending rate is not limited by the advertised receiver window, and the sender always has sufficient data to send. This assumption is satisfied easily by setting a large buffer size at the

- receiver. However, extension of the model to limited receiver window can also be included.
- Packet-losses in a round are independent of losses in other rounds. Here, a “round” is defined as the time between the sending of the first packet in a window to the receipt of the corresponding ACK. It is assumed that all packets will be sent after the first lost packet in a specific round will also be lost (same assumption as in PFTK model).
- The time required to send a window of data is smaller than RTT.
- The fluctuation of RTT measurements, in the absence of delay spikes, is assumed to compose a stationary random process with an expected value of RTT.
- Since the main concern in this thesis is to model the effect of packet-loss and long delays on TCP, SFR is prevented. Otherwise SFR usually follows a spurious timeouts. This is because the spuriously retransmitted packets produce a sequence of DUPACK at the receiver.

### 3.3 Basics of the PLLDC TCP Analytical Model

Before we proceed with the development of the proposed PLLDC model for estimating TCP Reno sending rate, throughput, and utilization factor in a noisy (error-prone) wireless environment and in presence of long delays, we introduce, first, the dynamics of sender window around a long delay, and then the statistical modeling of the long delay pattern.

#### 3.3.1 Dynamics of sender window around a long delay

In order to analyze the dynamics of the sender window around a long delay, it is important to be familiar with the evolution of sender’s window size as represented by the number of packets that can be sent. This is shown in Figure (3.6). As it has been described before, at each round the window is increased

by  $1/b$ , where  $b$  is number of packets acknowledged by one ACK packet, and  $b = 2$  when delayed ACK is used at the receiver. After  $X_i$  rounds, the long delay ( $D$ ) begin, when some of the packets in the  $X_i$ -th round are delayed (packets marked “d”) [Tek 08, Kli 08, Fu 03, Has 03].

Since the long delay is of a much larger timescale than a round, any extra packets that were sent in round  $X_{i+1}$ , corresponding to the ACKs of successfully delivered packets of round  $X_i$ , are also delayed. After  $T_o$  seconds, which is the converged value of the TO when the RTT is stable for a relatively long period

of time, the sender will TO and reduce the window to one and retransmit the first delayed packet. If it is not acknowledged within  $2T_o$ , the sender will retransmit it again, and so on.

The number of retransmissions during the long delay is denoted by  $R_D$ ; all these retransmitted packets are also delayed. Eventually, when the ACK for the first delayed packets comes back after the long delay has cleared, the sender will enter slow start and spuriously retransmit all the delayed packets. The sender will exit slow start when the window hits the slow start threshold, which is denoted as  $SST$ .

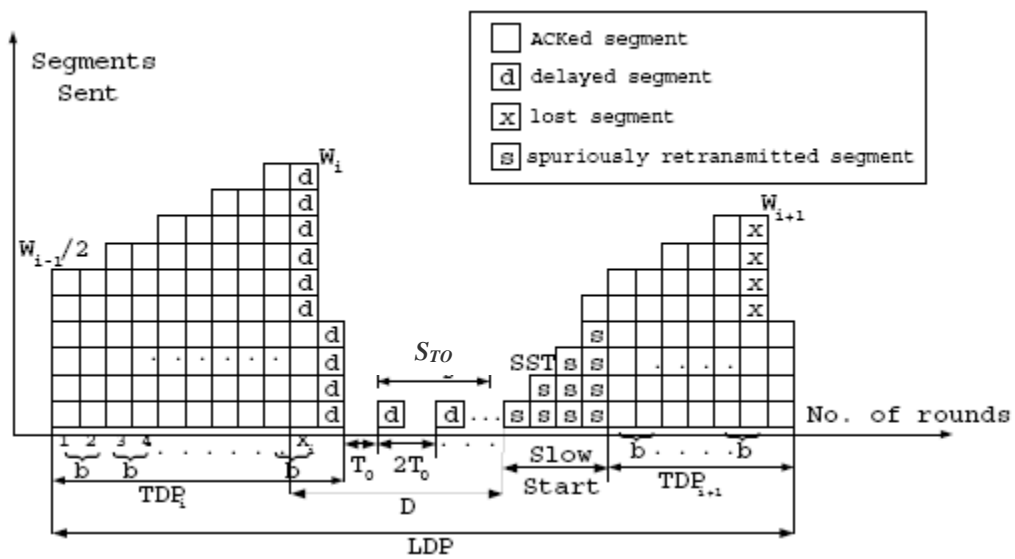


Figure (3.6): Packet sent during one Long Delay Period (LDP).

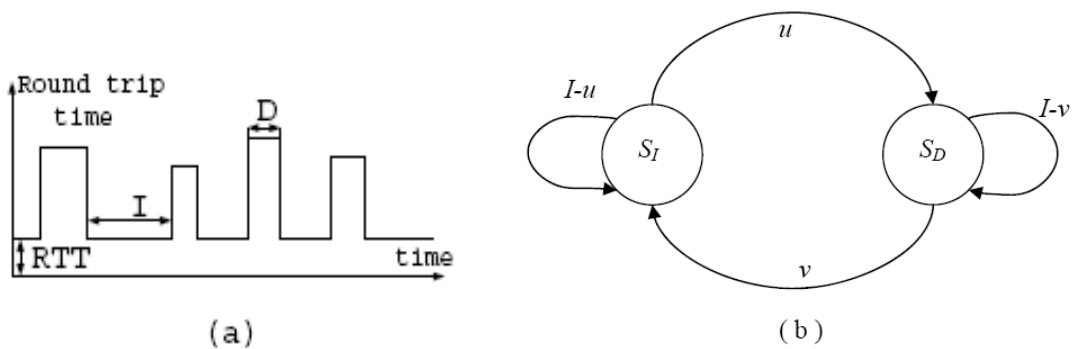


Figure (3.7). (a) Variation of *RTT* showing four long delays, and (b) model of long delays.



TCP Reno starts fast retransmit after receiving triple-DUPACK, which are called TD loss indications. The long delay period (LDP) is defined as consisting of two consecutive TDPs, one long delay, and one slow start as shown in Figure (3.6). TDP was defined in Section (3.1) as a period between two successive TD loss indications [Pad 00]. Note that even though the first period, labeled with  $TDP_i$  in the figure, does not end with a TD loss indication, the number of packets sent and the duration of  $TDP_i$  is the same as other TDPs, so just TDP will be used for convenience. The sender's window was  $W_{i-1}/2$  at the end of  $TDP_{i-1}$ ; after the fast retransmit, it has been reduced to  $W_{i-1}/2$ , which is the sender's window at the start of  $TDP_i$ .

### 3.3.2 Statistical modeling of the long delay pattern

It has been shown in Eqn. (1.1) that the sending rate is obtained by dividing the window size by the  $RTT$ , where  $RTT$  is the expected value of RTT when there is no long delay. However, in the presence of long delay spikes, the actual time is much more than RTT, and the time measured by the sender is referred to as the long delays as illustrated in Figure (3.7a) [Hou 08, Hes 05]. A two-state Markov chain is used to model the start and end of a long delay as shown in Figure (3.7b) [Alt 05]. The two states are: interval between long delays ( $S_I$ ) and duration of long delay ( $S_D$ ). Here, it is assumed that the length of the  $S_I$  and  $S_D$  states are both exponentially distributed, with  $u$  and  $v$  being the transition probabilities from state  $S_I$  to state  $S_D$  and state  $S_D$  to state  $S_I$ , respectively. By solving the Markov chain in Figure (3.7b), the relationship between  $I$  and  $D$  can be expressed as:

$$E[D] = \frac{u E[I]}{u + v} \quad (3.23)$$

Given a model for the lower layer events (such as link layer retransmission, mobile handoff, etc.) that cause long delays, the values of  $D$ ,  $I$ ,  $u$ , and  $v$  can be obtained to be used in Eqn. (3.23) [Kha 02].

### 3.4 Modeling of the TCP Reno Sending Rate (S) (PLLDC Model)

The sending rate is obtained by analyzing the dynamics of the sender's window around a long delay. In general, the sending rate is defined as the number of packets sent by a TCP source per unit time, or the rate at which packets sent by a TCP source. In wireless environment, due to packet-loss and presence of long delays; the number of packets sent by a TCP source over a short period of time is changing continuously. Therefore, to estimate a satisfactory value for sending rate an adequate period of time needs to be considered. It will be more advantageous if the duration of the monitoring period is dynamically adjusted depending on the packet-error rate [Tek 08, Hes 05, Yan 03].

In order to develop a satisfactory mathematical model for estimating sending rate of wireless TCP source over an unreliable link, with certain packet-error rate ( $p$ ), and in the presence of long delay spikes, the sending rate of the TCP source can be expressed as:

$$E[S] = \frac{E[S_{LDC}]}{E[D_{LDC}]} = \frac{mE[S_{NP}] + E[S_{LDP}]}{mE[D_{NP}] + E[D_{LDP}]} \quad (3.24)$$

Where  $S$  sending rate of the TCP source

$S_{LDC}$  number of packets sent during one long delay cycle (LDC)

$D_{LDC}$  duration of a long delay cycle (LDC)

$m$  number of normal periods (NPs) in one LDC

$S_{NP}$  number of packets sent during one NP

$S_{LDP}$  number of packets sent during one long delay period (LDP)

$D_{NP}$  duration of one NP

$D_{LDP}$  duration of one long delay period (LDP)

In the above equation, the numerator denotes the number of packets sent during one LDC and the denominator is the duration of an LDC. We first look at the macroscopic behavior of one LDP, which will then be used to determine the number of packets sent and the duration of an LDC.

### 3.4.1 Analysis of a long delay period (LDP)

The total number of packets sent during one LDP is the sum of packets sent during two TDPs, the TO period, and the slow start stage as shown in Figure (3.6):

$$E[S_{LDP}] = 2E[S_{TDP}] + E[R_D] + E[S_{SST}] \quad (3.25)$$

The duration of LDP can be written as the sum of the time duration of the two TDP periods, the long delay, and one slow start stage, minus the overlapping area ( $2RTT$ ) between  $D$  and TDP:

$$E[D_{LDP}] = 2E[D_{TDP}] + E[D] + k RTT - 2 RTT \quad (3.26)$$

$$E[D_{LDP}] = 2E[D_{TDP}] + E[D] + (k - 2) RTT \quad (3.27)$$

$E[S_{TDP}]$  and  $E[D_{TDP}]$  in Eqns. (3.25) and (3.27) can be determined from Eqns. (3.2) and (3.3), respectively. Next, the three unknown variables  $E[R_D]$  and  $E[S_{SST}]$  in Eqn. (3.25), and  $k$  in Eqn. (3.27) are derived below.

#### i. Deriving $E[R_D]$

Since  $D$  is assumed to be exponentially distributed with mean  $E[D]$ , and the sender experiences a long delay of  $D$ , then the probability that there is one TO can be expressed as:

$$Pr(T_o < D \leq 2T_o) = Pr(D \leq 2T_o) - Pr(D \leq T_o) \quad (3.28)$$

$$Pr(T_o < D \leq 2T_o) = e^{-\frac{T_o}{E[D]}} - e^{-\frac{2T_o}{E[D]}}$$

(3.29)

The probability that there are two or more TOs is:

$$Pr(D \leq 2T_o) = e^{-\frac{2T_o}{E[D]}} \quad (3.30)$$

Because the sender sends out a packet when a TO occurs, the number of packets sent during  $D$  is the same as the number of TOs. Since the sender can back-off a maximum of 6 times to get a TO of  $64T_o$ , the number of packets sent can be expressed as:

$$E[R_D] = Pr(T_o < D \leq 2T_o) + 2Pr(2T_o < D \leq 3T_o) + \dots + 6Pr(32T_o < D \leq 64T_o)$$

$$E[R_D] = \sum_{j=0}^5 \left( e^{-\frac{2^j T_o}{E[D]}} - e^{-\frac{64T_o}{E[D]}} \right) \quad (3.31)$$

## ii. Deriving $k$

After the long delay, the SST value will be  $\max(W_i/2, 2)$  if there is only one TO during  $D$ , otherwise, it will be two for two or more TOs. Therefore, the expected value of SST after the long delay is:

$$E[SST] = \max\left(\frac{W_i}{2}, 2\right) \left( e^{-\frac{T_o}{E[D]}} - e^{-\frac{2T_o}{E[D]}} \right) + 2e^{-\frac{2T_o}{E[D]}} \quad (3.32)$$

During the slow start, if the receiver adopts delayed acknowledgment, the sender's congestion window will grow by half of the window size in the previous round according to the following rule:

$$cwnd_{j+1} = cwnd_j + \left\lceil \left( \frac{cwnd_j}{2} \right) \right\rceil \quad \text{with } cwnd_j=1 \text{ and } j = 1, 2, 3, \dots$$

This can be approximated as:

$$cwnd_j = \left(\frac{3}{2}\right)^j \text{ and } j = 1, 2, 3 \dots \quad (3.33)$$

End of the slow start stage at  $E[SST]$  after  $k$  rounds implies that  $cwnd_k = E[SST]$ ; the number of rounds needed to complete this stage is approximately expressed as:

$$k = \left\lceil \frac{\ln(E[SST])}{\ln(1.5)} \right\rceil \quad (3.34)$$

### iii. Deriving $E[S_{SST}]$

The number of packets sent in each round of the slow start stage in Figure (3.6) is given in Eqn. (3.33). So the number of packets sent during slow start can be approximated by the sum of the packets sent during these  $k$  rounds:

$$E[S_{SST}] = \sum_{j=1}^k \left(\frac{3}{2}\right)^j \approx 3 \left(\frac{3}{2}\right)^k - 3 \quad (3.35)$$

By substituting  $E[R_D]$ ,  $k$ , and  $E[S_{SST}]$  from Eqns. (3.31), (3.33), and (3.35) into Eqns. (3.25) and (3.27), we can obtain the number of packets sent and the duration of one LDP.

## 3.4.2 Analysis of one long delay cycle (LDC)

It can be seen from Figure (3.8) that the total number of packets sent during one LDC ( $S_{LDC}$ ) is the sum of packets sent during  $m$  instances of NP periods and an LDP period. Thus,  $E[S_{LDC}]$  can be expressed as:

$$E[S_{LDC}] = m E[S_{NP}] + E[S_{LDP}] \quad (3.36)$$

Where  $E[S_{LDP}]$  represents the total number of packets sent during one LDP and it can be calculated using Eqn. (3.25), and  $E[S_{NP}]$  represents the number of packets sent during the  $i^{\text{th}}$  NP ( $i=1, 2, 3, \dots, m$ ), which can be obtained from [Pad 00] as given below:

$$E[S_{NP}] = \frac{1}{Q} E[S_{TDP}] + E[R_{NP}] \quad (3.37)$$

Where  $E[S_{TDP}]$ ,  $Q$ , and  $E[R_{NP}]$  can be determined using Eqns. (3.2), (3.8), and (3.10), respectively. Substituting these equations for Eqn. (3.37) yields the following expression for  $E[S_{NP}]$ :

$$E[S_{NP}] = \frac{1}{\min(1, \frac{3}{E[W]})} \left( \frac{1-p}{p} + \frac{2+b}{3b} + \sqrt{\frac{8(1-p)}{3bp} + \left(\frac{2+b}{3b}\right)^2} \right) + \frac{1}{1-p} \quad (3.38)$$

Where  $E[W]$  can be obtained from Eqn. (3.9). In Eqn. (3.24),  $E[D_{LDP}]$  has already been developed in Eqn. (3.27). Another parameter in Eqn. (3.24) that needs to be determined is  $E[D_{NP}]$ , which denotes the duration of the  $i^{\text{th}}$  NP ( $i=1, 2, 3, \dots, m$ ). However, for  $E[D_{NP}]$  an expression was derived and can be obtained from [Pad 00]. It is given as:

$$E[D_{NP}] = \frac{1}{Q} E[D_{TDP}] + E[D_{TO}] \quad (3.39)$$

Where  $E[D_{TDP}]$ ,  $Q$ , and  $E[D_{TO}]$  can be determined using Eqns. (3.3), (3.8), and (3.11), respectively. Substituting these equations for Eqn. (3.39) yields the following expression for  $E[D_{NP}]$ :

$$E[D_{NP}] = \frac{RTT}{\min(1, \frac{3}{E[W]})} \left( \frac{2+b}{6} + \sqrt{\frac{2b(1-p)}{3p} + \left(\frac{2+b}{6}\right)^2} + 1 \right) + T_o \frac{f(p)}{1-p} \quad (3.40)$$

Once again,  $E[W]$  and  $f(p)$  can be obtained from Eqns. (3.9) and (3.12), respectively. At this stage, to estimate the sending rate using Eqn. (3.24), only parameter remains and needs to be determined which is  $m$ . In order to derive an expression for determining  $m$ , let us go back to the definition of LDC in Figure (3.8). It starts with the end of the previous LDP. An LDC consists of several instances of NPs at the beginning and an LDP at the end. In the new model, the normal period,  $E[D_{NP}]$ , denotes the time interval with no long delays, which is given by Eqn. (3.40). It is equal to the sum of  $D_{TD}$  and  $D_{TO}$ .

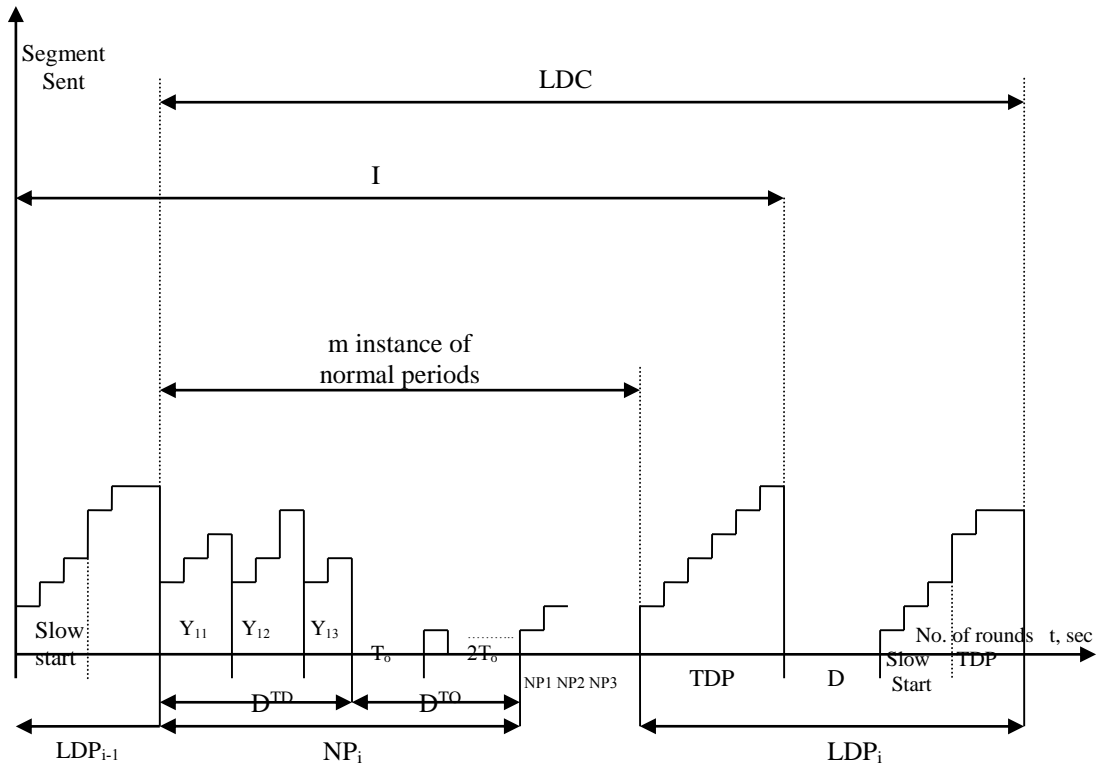


Figure (3.8): Sender window evaluation in one LDC.

Referring to Figure (3.8) the interval between long delays ( $I$ ) consists of a slow start phase following the previous long delay,  $m$  instances of NP and a TDP. We can calculate  $m$  as:

$$m = \frac{E[I] - 2E[D_{TDP}] - k \cdot RTT}{E[D_{NP}]} \quad (3.41)$$

As it has been stated earlier that the total number of packets sent during one LDC is the sum of packets sent during  $m$  instances of NP period and an LDP period:

$$E[S_{LDC}] = \sum_{i=1}^m (S_{NP})_i + E[S_{LDP}] \quad (3.42)$$

Similarly, since one LDC consists of  $m$  instances of NP and ends with one LDP, the duration of one LDC can be obtained as:

$$E[D_{LDC}] = \sum_{i=1}^m (D_{NP})_i + E[D_{LDP}] \quad (3.43)$$

Due to the randomness of the process, the above two equations can be written in expected forms as follows:

$$E[S_{LDC}] = m E[S_{NP}] + E[S_{LDP}] \quad (3.44)$$

$$E[D_{LDC}] = m E[D_{NP}] + E[D_{LDP}] \quad (3.45)$$

By substituting  $E[S_{LDC}]$  from Eqn. (3.44), and  $E[D_{LDC}]$  from Eqn. (3.45) for Eqn. (3.24), the long-term steady-state sending rate of the TCP sender can be determined in presence packet-loss and long delay spikes. The packet-loss depends on the packet-error rate ( $p$ ), and the duration of the delay spikes depends on the duration of the long delay ( $D$ ), the interval between long delays ( $I$ ), and number of NPs in one LDC ( $m$ ). Thus,  $E[S]$  can be expressed as:

$$E[S] = \frac{E[S_{LDC}]}{E[D_{LDC}]} = \frac{mE[S_{NP}] + E[S_{LDP}]}{mE[D_{NP}] + E[D_{LDP}]} \quad (3.46)$$

### 3.5 Modeling of the TCP Reno Throughput ( $T$ ) (PLLDC Model)

In wireless communication networks, throughput is defined as the amount of digital data per time unit that is delivered over a physical or logical link, or that is passing through a certain network node [Lee 08, Pap 07, Hes 05]. For example, it may be the amount of data that is delivered to a certain network terminal or host computer, or between two specific computers. The throughput is usually measured in bit per second (bit/sec or bps). It is also measured in data packets per second or data packets per timeslot. The throughput gives indication on or corresponds to digital bandwidth consumption.

The maximum throughput of a node or communication link is synonymous to its capacity. The maximum throughputs is equal to or lower than the sending rate of a physical link, excluding physical layer protocol overhead such as channel coding [Kim 07].



In a wireless computer network, the throughput that is achieved from one computer to another may be lower than the maximum throughput, and than the network access channel capacity, for several reasons, for example [Che 08, Fu 03, Gur 01]:

- The traffic load may be lower than the maximum throughput.
- The channel capacity may be shared by other users. If a bottleneck communication link physical data rate  $C$  is shared by  $N$  users, every user typically achieves a throughput of approximately  $C/N$  if fair queuing best-effort communication is assumed.
- Flow control, for example in the TCP protocol, affects the throughput if the bandwidth-delay product is larger than the TCP window, i.e. the buffer size. In that case the sending computer must wait for acknowledgement of the data packets before it can send more packets.
- Packet-loss due to network congestion. Packets may be dropped in switches and routers when the packet queues are full due to congestion.
- Packet-loss due to bit errors.
- TCP congestion avoidance controls the data rate that occurs in the beginning of a file, and after packet drops caused by router congestion or bit errors in, for example, wireless links.
- Scheduling algorithms in routers and switches. If fair queuing is not provided, users that send large packet will get higher bandwidth. Some users may be prioritized in a weighted fair queuing (WFQ) algorithm if differentiated or guaranteed quality of service (QoS) is provided.
- Backoff waiting time after collisions.

The TCP throughput can be determined by subtracting the spuriously retransmitted and lost packets from the sending rate as in Figure (3.6), the delayed packets in the  $X_i$  and  $X_{i+1}$ -th rounds of the first TDP are subsequently

spuriously retransmitted, therefore, one window of packets  $E[W]$  must be subtracted from  $E[S_{TDP}]$ . So that the net number of packets send during the first TDP,  $E[S'_{TDP,1}]$ , can be expressed as:

$$E[S'_{TDP,1}] = E[S_{TDP}] - E[W] \quad (3.47)$$

Substituting  $E[S_{TDP}]$  and  $E[W]$  from Eqns. (3.2) and (3.9) for Eqn. (3.47) yields the following value for  $E[S'_{TDP,1}]$ :

$$E[S'_{TDP,1}] = \frac{1-p}{p} \quad (3.48)$$

In the second TDP of the LDP period, the lost and the delayed packets need to be subtracted from the sending rate, i.e., on the average,  $E[W]/2$  must be subtracted. So that the net number of packets send during the second TDP,  $E[S'_{TDP,2}]$ , can be expressed as:

$$E[S'_{TDP,2}] = E[S_{TDP}] - \frac{E(W)}{2} \quad (3.49)$$

Once again, substituting  $E[S_{TDP}]$  and  $E[W]$  from Eqns. (3.2) and (3.9) for Eqn. (3.49) yields the following value for  $E[S'_{TDP,2}]$ :

$$E[S'_{TDP,2}] = \frac{1-p}{p} + \frac{1}{2} \left( \frac{2+b}{3b} + \sqrt{\frac{8(1-p)}{3bp} + \left(\frac{2+b}{3b}\right)^2} \right) \quad (3.50)$$

Because the packets retransmitted during the TO period are discarded by the receiver,  $E[R_{NP}]$  in Eqn. (3.37) can be replaced with  $E[R'_{NP}] = 1$ . Similarly,  $E[R_D]$  in Eqn. (3.25) can be replaced with  $E[R'_D] = 1$ . Replacing  $E[S_{TDP}]$ ,  $E[R_{NP}]$  and  $E[R_D]$  in Eqns. (3.25) and (3.37) with  $E[S'_{TDP,1}]$ ,  $E[S'_{TDP,2}]$ ,  $E[R'_{NP}]$  and  $E[R'_D]$ , the following expressions can be found for  $E[S'_{LDP}]$  and  $E[S'_{NP}]$ :

$$E[S'_{LDP}] = E[S'_{TDP,1}] + E[R'_D] + E[S_{SST}] + E[S'_{TDP,2}] \quad (3.51)$$

$$E[S'_{NP}] = \frac{1}{Q} E[S'_{TDP,2}] + E[R'_{NP}] \quad (3.52)$$

Therefore, the average TCP throughput during one LDC can be calculated as the total number of packets delivered to the receiver  $E[S'_{LDC}]$  divided by the duration of one LDC  $E[D_{LDC}]$ .  $E[S'_{LDC}]$  can be obtained by replacing  $E[S_{LDP}]$  and  $E[S_{NP}]$  in Eqn. (3.44) with  $E[S'_{LDP}]$  and  $E[S'_{NP}]$ , So that

$$E[S'_{LDC}] = m E[S'_{NP}] + E[S'_{LDP}] \quad (3.53)$$

Although, the spuriously retransmitted and lost packets are subtracted from the total number of packets received at the receiver, the duration of an LDC remains unchanged. Therefore, the throughput ( $T$ ) of the TCP connection can be determined as:

$$E[T] = \frac{E[S'_{LDC}]}{E[D_{LDC}]} = \frac{m E[S'_{NP}] + E[S'_{LDP}]}{m E[D_{NP}] + E[D_{LDP}]} \quad (3.54)$$

Where  $E[S'_{LDC}]$  is given in Eqn. (3.53) and it represents the actual number packets delivered to the receiver, and  $E[D_{LDC}]$  is given in Eqn. (3.45) and it represents the duration of one LDC.

It is not recommended to measure throughput in percentage, to avoid confusion regarding what the percentage is related to. It is better to use the utilization factor and drop rate in percentage.

### 3.6 Modeling of the TCP Reno Utilization Factor ( $U$ ) (PLLDC Model)

The utilization factor ( $U$ ) in percentage is determined by dividing the achieved throughput ( $T$ ) by the sending rate ( $S$ ) of a TCP source. Thus, the utilization factor ( $U$ ) can be expressed mathematically as:

$$U = \frac{T}{S} \quad (3.55)$$

Where  $T$  is the throughput and  $S$  is the sending rate. Substituting  $T$  and  $S$  from Eqns. (3.54) and (3.46) for Eqn. (3.55),  $U$  can be expressed as:

$$E[U] = \frac{E[T]}{E[S]} \times 100 = \frac{m E[S'_{NP}] + E[S'_{LDP}]}{m E[S_{NP}] + E[S_{LDP}]} \times 100 \quad (3.56)$$

Where  $m$ ,  $E[S'_{NP}]$ ,  $E[S'_{LDP}]$ ,  $E[S_{NP}]$ , and  $E[S_{LDP}]$  are given by Eqns. (3.41), (3.52), (3.51), (3.38), and (3.27), respectively.

### 3.7 Implementations

The PLLDC model is implemented in C++ code. The input parameters are listed and defined in Table (3.1), while the code main computed parameters are given in Table (3.2). Each parameter is computed by the equation indicated in column #3 in Table (3.2). The computed parameters are ordered in the sequence by which they are computed. They are categorized into four main groups, these are:

1. Group 1: Parameters represent length of time periods or durations.
2. Group 2: Parameters represent the total number of packets sent during all durations, which are required for computing the TCP sending rate.
3. Group 3: Parameters represent the net number of packets sent during all durations, which are required for computing the TCP throughput.
4. Group 4: Main computed parameters (e.g., TCP sending rate, throughput, and utilization factor).

Table (3.1) Definition of the code input parameters.	
Sym.	Description
$p$	Packet error rate.
$D$	Duration of the long delay.
$I$	Interval between long delays.
$T_0$	Timeout (TO)
$SST$	Value of slow start threshold at the end of a long delay $D$ .
$b$	Number of packets acknowledged by one ACK packet. $b = 2$ when delayed acknowledgment is used at the receiver.
$RTT$	Expected value of round trip time (RTT) when there is no long delay.
$W_m$	The receiver's maximum advertised congestion window size.

Table (3.2) Definition of the code computed parameters.		
Sym.	Description	Equation
Length of each duration		
$D_{TDP}$	Duration of triple duplicate period (TDP), i.e. the time between two successive triple duplicate loss indications.	(3.3)
$f(p)$	Polynomial function.	(3.12)
$D_{TO}$	Duration of the TO period in one normal point (NP).	(3.11)
$W$	TCP sender window size.	(3.9)
$Q$	The probability that a loss indication in a window of size $W$ is TO.	(3.8)
$Q_m$	The probability that a loss indication in a window of size $W_m$ is TO.	(3.15)
$D_{NP}$	Duration of one NP, where NP consists of $n$ instances of TDP and one instance of TO period.	(3.39)
$k$	The number of rounds needed to complete the slow start stage after a long delay.	(3.34)
$m$	Number of NPs in one long delay cycle (LDC).	(3.41)
$D_{LDP}$	Duration of long delay period (LDP), which consists of one TDP, one long delay, one slow start, and a second TDP.	(3.27)
$D_{LDC}$	Duration of LDC, which consists of $m$ NPs and one LDP.	(3.43)
Number of packets sent during each duration		
$S_{TDP}$	Number of packets sent from the sender during one TDP.	(3.2)
$R_{NP}$	Number of packets sent during the TO period in one NP.	(3.10)
$S_{NP}$	Number of packets sent during $i^{th}$ NP, $i = 1, 2, \dots, m$ .	(3.38)
$R_D$	Number of packets sent during long delay ( $D$ ).	(3.31)
$S_{SST}$	Number of packets sent during the slow start stage in an LDP.	(3.35)
$S_{LDP}$	Number of packets sent during one LDP.	(3.25)

$S_{LDC}$	Number of packets sent during one LDC.	(3.44)
Net number of packets sent during each duration		
$S'_{TDP,1}$	Net number of packets sent during the first TDP.	(3.48)
$S'_{TDP,2}$	Net number of packets sent during the second TDP.	(3.49)
$R'_D$	Number of packets sent during long delay ( $R'_D = 1$ ).	-
$R'_{NP}$	Number of packets sent during the TO period in one NP ( $R'_{NP} = 1$ ).	-
$S'_{LDP}$	Net number of packets sent during one LDP.	(3.51)
$S'_{NP}$	Net number of packets sent during $i^{th}$ NP, $i = 1, 2, \dots, m$ .	(3.52)
$S'_{LDC}$	Net number of packets delivered to the receiver during one LDC.	(3.53)
Main computed parameters		
$S$	Long-term steady-state sending rate of TCP connection.	(3.46)
$T$	Long-term steady-state throughput of TCP connection.	(3.54)
$U$	Long-term steady-state utilization factor.	(3.56)

## Chapter 4

### Results and Discussions

Chapter 3 presented a detail description and derivation of a new mathematical model for evaluating the performance of TCP Reno in a wireless environment that suffers from wide range of packet loss (PL) and long delay cycles (LDCs). Therefore, we referred to it as PLLDC model. The performance of the TCP Reno is mainly evaluated in terms of the following parameters:

- i. Sending rate ( $S$ )
- ii. Throughput ( $T$ )
- iii. Utilization factor ( $U$ )

In order to assess the effectiveness of the new model in evaluating the performance of TCP Reno in a realistic wireless environment that suffers from Packet-Loss and Long Delay Cycles, it is implemented in C++ code, and used to predict the performance in three different wireless network environments (scenarios). These three scenarios evaluate the performance of TCP Reno by investigating the variation of  $S$ ,  $T$ , and  $U$  with the Packet-Loss rate ( $p$ ) for various values of Long Delays ( $D$ ), Interval between Long Delays ( $I$ ), and Round Trip Time ( $RTT$ ), respectively.

The main objectives of these three scenarios can be summarized as follows:

- i. Scenario #1: Investigate the effect of Long Delays ( $D$ ).
- ii. Scenario #2: Investigate the effect of Interval between Long Delays ( $I$ ).
- iii. Scenario #3: Investigate the effect of Round Trip Time ( $RTT$ ).

Also, in order to provide an insight into the behavior of TCP Reno in various wireless network environments, the effect of a number of parameters are investigated. These parameters include:

- i. Number of packets sent during one LDC ( $S_{LDC}$ )
- ii. Number of packets sent during one LDP ( $S_{LDP}$ )
- iii. Number of packets sent during one NP ( $S_{NP}$ )
- iv. Number of packets sent during one TDP ( $S_{TDP}$ )
- v. Number of packets sent during the timeout period in one NP ( $R_{NP}$ )
- vi. Number of packets sent during long delay ( $R_D$ )
- vii. Number of packets sent during the slow start stage in one LDP ( $S_{SST}$ )

All results obtained from our PLLDC model are validated against results obtained from the well-known PFTK model [Pad 00] and the widely-used NS-2 network simulator [Fu 03, Has 03]. Since, the main objective of using NS-2 is for validating the accuracy of our new model, results from NS-2 will be compared in Scenario #1 only. This is because the NS-2 simulation results show an excellent agreement with our analytical PLLDC model for values of  $S$ ,  $T$ , and consequently  $U$ .

The results obtained for these three scenarios are presented in tables and graphs. The results are plotted in semi-logarithmic or logarithmic scale to accommodate the wide ranges of  $p$  and the predicted parameters. Finally, in this chapter, the results obtained are discussed.



## 4.1 Scenario #1: Investigates the Effect of Long Delays ( $D$ )

This scenario investigates the effect of  $D$  on the performance of TCP Reno. The performance of TCP Reno is investigated by using the PLLDC model to predict the variation of  $S$ ,  $T$ , and  $U$  with  $p$  for various values of  $D$ . All other input parameters remain unchanged during this scenario, and the list and values of the input parameters for Scenario #1 are summarized in Table (4.1).

Table (4.1) Input parameter for Scenario #1	
Parameter	Value
Packet-loss rate ( $p$ )	0.001 to 0.5
Duration of the long delay ( $D$ )	6, 8, 10, 12 sec
Interval between long delays ( $I$ )	30 sec
Round Trip Time ( $RTT$ )	0.200 sec
Retransmission timeout or timeout ( $T_o$ )	1.00 sec
Slow start threshold at the end of a long delay $D$ ( $SST$ )	2
Number of packets acknowledged by one ACK packet ( $b$ ).	2
Receiver's maximum congestion window size ( $W_m$ )	800 packet

### 4.1.1 Scenario #1: Sending rate ( $S$ )

The PLLDC model is used to predict the variation of  $S$  with  $p$  for four various values of  $D$  (6, 8, 10, and 12 sec). The results obtained are listed in Table (4.2) and plotted in Figure (4.1). Figure (4.1) shows that, as expected,  $S$  is inversely proportional to  $p$ , i.e., it decreases as  $p$  increases for all values of  $D$ . Furthermore, if all other network parameters remain unchanged,  $S$  slightly decreases as  $D$  increases, in other word;  $D$  only has a slight effect on  $S$ .

The results demonstrate that when  $D$  is doubled,  $S$  is only decreased by approximately 14% for all values of  $p$ . For example, as shown in Table (4.2), when  $p=0.01$  and  $D$  is doubled (increased from 6 to 12 sec),  $S$  is reduced by 14.5% (reduced from 32.80 to 28.09 packets/sec). The values in brackets

shown in Table (4.2) are referred to the percentage decrease in  $S$  as compared to the values of  $S$  when  $D=6$  sec. The negative sign indicates that  $S$  decreases as  $D$  increases.

In order to validate the accuracy of the PLLDC model, the sending rates predicted by the PLLDC model are compared with sending rates predicted by the PFTK model and NS-2 network simulator. A numerical comparison is presented in Table (4.2) and plotted in Figure (4.2). The figure shows that the PLLDC model can predict  $S$  more accurately than the PFTK model. It is also shown that when  $D$  increases, the gap between the PFTK model and the simulation result increases, but the PLLDC model accommodates the increase of  $D$  well.

Table (4.2) Comparison of sending rate ( $S$ ) for Scenario #1.										
#	$\rho$	PFTK model	Sending rate ( $S$ )							
			$D$							
			6		8		10		12	
			PLLDC	NS-2	PLLDC	NS-2	PLLDC	NS-2	PLLDC	NS-2
1	0.001	134.32	112.46	125.2 7	106.49 (-5.3)*	116.2 2	101.12 (-10.1)	107.46	96.26 (-14.4)	100.2 7
2	0.005	57.30	48.38	54.51	45.49 (-6.0)	48.91	43.51 (-10.1)	44.87	41.42 (-14.4)	42.47
3	0.01	38.60	32.80	33.22	31.07 (-5.3)	32.37	29.50 (-10.1)	30.50	28.09 (-14.4)	29.03
4	0.05	13.11	11.47	11.02	10.87 (-5.2)	10.66	10.33 (-10.0)	10.07	9.83 (-14.3)	9.88
5	0.1	7.24	6.51	6.66	6.17 (-5.2)	6.35	5.87 (-9.8)	5.92	5.59 (-14.1)	5.81
6	0.5	0.58	0.81	-	0.77 (-4.9)	-	0.74 (-8.6)	-	0.71 (-12.4)	-

\* The values in brackets represent the percentage decrease in  $S$  as compared to the values of  $S$  when  $D=6$  sec. The negative sign indicates that  $S$  decreases as  $D$  increases.

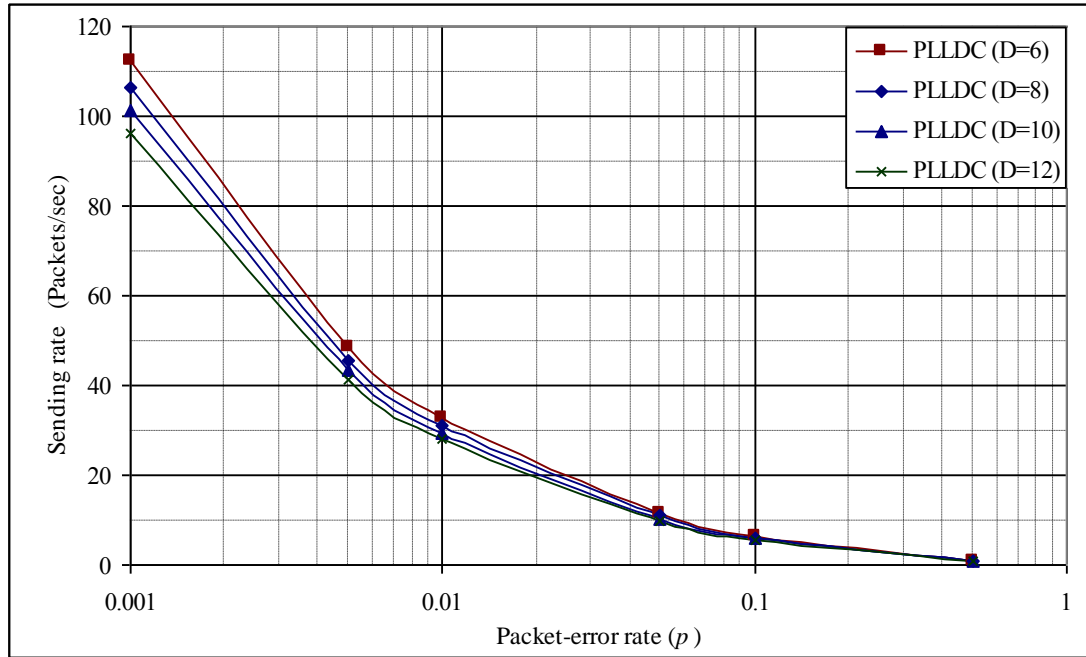


Figure (4.1). Variation of  $S$  with  $p$  for various values of  $D$  for Scenario #1.

In order to provide an insight into the behavior of TCP Reno in a noisy environment and during long delay cycles, Table (4.3) lists the values of the number of packets sent during each time period within the long delay cycle, namely,  $S_{LDC}$ ,  $S_{LDP}$ ,  $S_{NP}$ ,  $S_{TDP}$ ,  $R_{NP}$ ,  $R_D$ , and  $S_{SST}$ . Definitions of these parameters were given in Chapter 3. The values presented in Table (4.3) are plotted in Figure (4.3)

It can be seen from the results in Table (4.3) and Figure (4.3) that  $S_{NP}$ ,  $S_{TDP}$ ,  $R_{NP}$ , and  $S_{SST}$  are not affected by the variation of  $D$  and only vary with  $p$ . This can be clarified by looking back to Eqns. (3.38), (3.2), (3.10), and (3.35), respectively. The value of  $R_D$  (number of retransmitted packets during  $D$ ) is increasing with  $D$ ; consequently, the values of  $S_{LDC}$  (Eqn. (3.44)) and  $S_{LDP}$  (Eqn. (3.25)) are also increased.

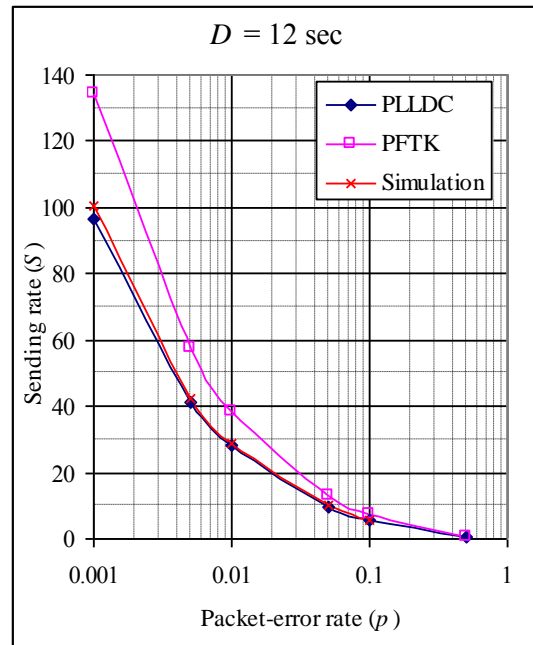
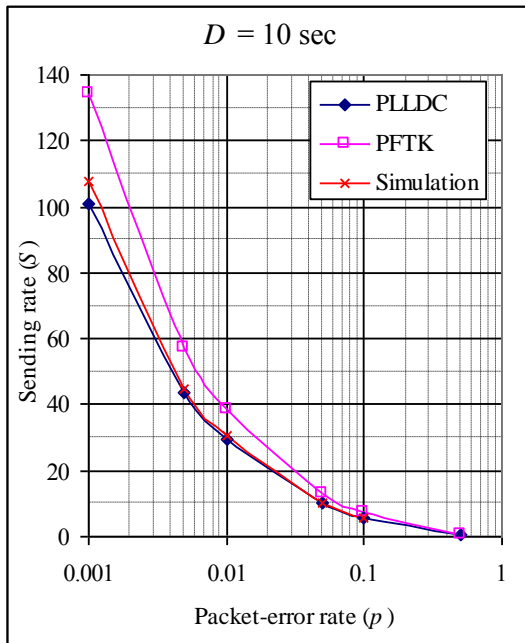
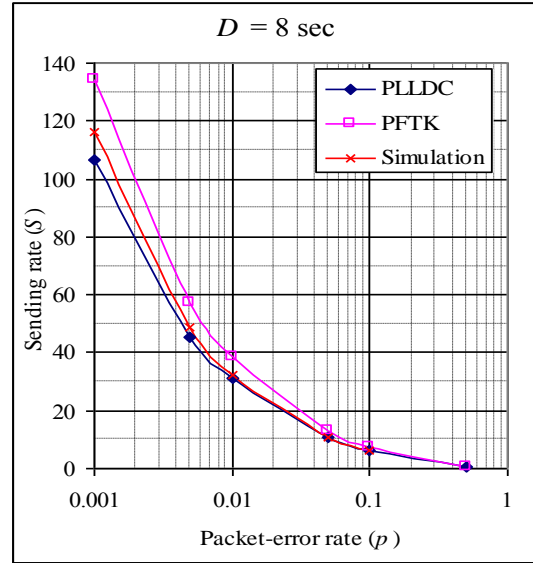
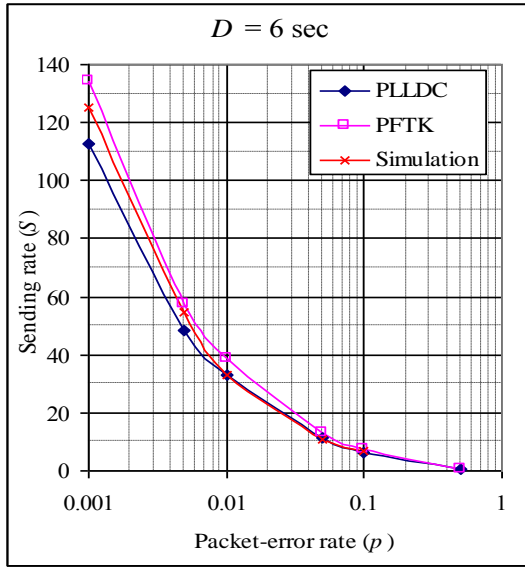


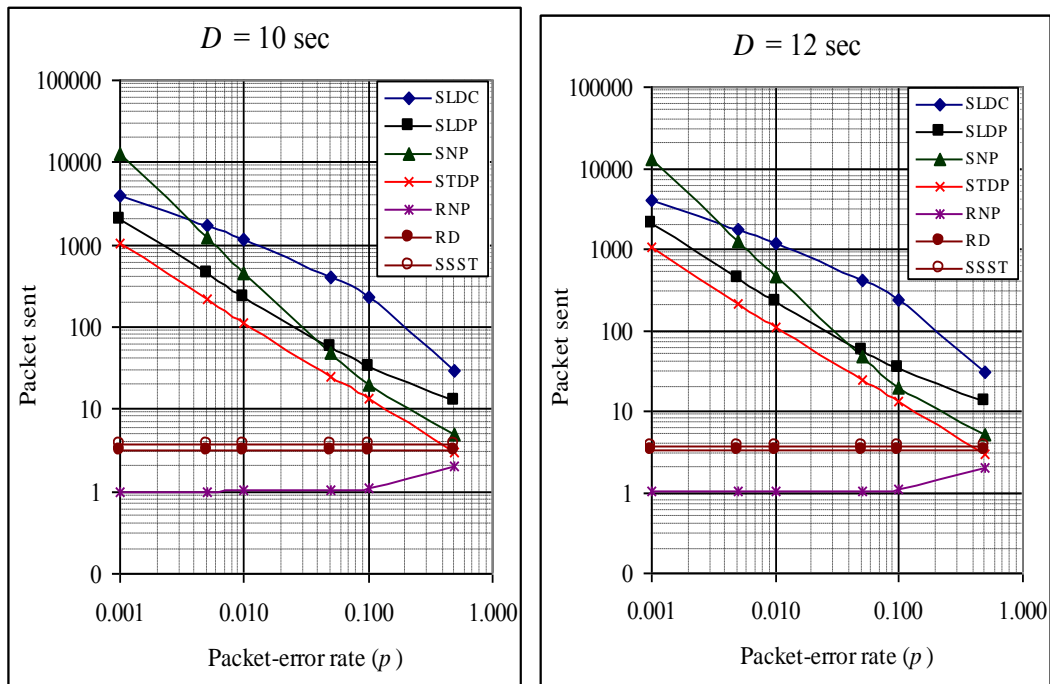
Figure (4.2). Comparison of S for Scenario #1.

Table (4.3) Number of packets sent during the different time periods of a long delay cycle.							
$p$	$S_{LDC}$	$S_{LDP}$	$S_{NP}$	$S_{TDP}$	$R_{NP}$	$R_D$	$S_{SST}$
$D = 6$							
0.001	4003.62	2078.50	12838.90	1036.17	1.00	2.41	3.75
0.005	1722.24	438.10	1222.63	215.97	1.01	2.41	3.75
0.01	1167.71	228.51	452.20	111.18	1.01	2.41	3.75
0.05	408.39	55.65	48.43	24.74	1.05	2.41	3.75
0.1	231.74	32.55	19.56	13.19	1.11	2.41	3.75
0.5	28.68	12.16	5.00	3.00	2.00	2.41	3.75
$D = 8$							
0.001	4004.00	2078.88	12838.90	1036.17	1.00	2.79	3.75
0.005	1722.61	438.48	1222.63	215.97	1.01	2.79	3.75
0.01	1168.08	228.89	452.20	111.18	1.01	2.79	3.75
0.05	408.76	56.03	48.43	24.74	1.05	2.79	3.75
0.1	232.11	32.93	19.56	13.19	1.11	2.79	3.75
0.5	29.05	12.54	5.00	3.00	2.00	2.79	3.75
$D = 10$							
0.001	4004.29	2079.16	12838.90	1036.17	1.00	3.08	3.75
0.005	1722.90	438.77	1222.63	215.97	1.01	3.08	3.75
0.01	1168.37	229.18	452.20	111.18	1.01	3.08	3.75
0.05	409.05	56.31	48.43	24.74	1.05	3.08	3.75
0.1	232.40	33.21	19.56	13.19	1.11	3.08	3.75
0.5	29.34	12.83	5.00	3.00	2.00	3.08	3.75
$D = 12$							
0.001	4004.51	2079.39	12838.90	1036.17	1.00	3.30	3.75
0.005	1723.13	438.99	1222.63	215.97	1.01	3.30	3.75
0.01	1168.59	229.40	452.20	111.18	1.01	3.30	3.75
0.05	409.27	56.54	48.43	24.74	1.05	3.30	3.75
0.1	232.63	33.44	19.56	13.19	1.11	3.30	3.75
0.5	29.56	13.05	5.00	3.00	2.00	3.30	3.75

Figure (4.3). Packets send during the different stages of a Long Delay Cycle for various values of  $D$  for Scenario #1.

### 4.1.2 Scenario #1: Throughput ( $T$ )

The PLLDC model is also used to predict the variation of  $T$  with  $p$  for four various values of  $D$  (6, 8, 10, and 12 sec). The results obtained are listed in Table (4.4) and plotted in Figure (4.4). Once again  $T$  is inversely proportional to  $p$ , and if all other network parameters remain unchanged,  $T$  slightly decreases as  $D$  increases. As for  $S$ ,  $T$  is reduced by nearly 14% if  $D$  is doubled, for all values of  $p$ . For example, as shown in Table (4.4), when  $p=0.01$  and  $D$  is doubled (increases from 6 to 12 sec),  $T$  is reduced by 14.44% (reduced from 30.81 to 26.36 packets/sec). The values between brackets shown in Table (4.4)



represent the percentage decrease in  $T$  as compared to the values of  $T$  when  $D=6$  sec. The negative sign indicates that  $T$  is inversely proportional to  $D$ .

The variation (percentage increases or decreases) is calculated according to the following formula:

$$V = \frac{X_2 - X_1}{X_1} \times 100 \quad (4.1)$$

Where  $V$  represents the variation with respect to a reference value ( $X_1$ ). A positive value of  $V$  refers to an increase with respect to  $X_1$ , while a negative refers to a decrease with respect to  $X_1$ .  $X_1$  is the reference value (in the above example, the value of  $S$  at  $p=0.01$  and  $D=6$  sec).  $X_2$  is the compared value (in the above example, the value of  $S$  at  $p=0.01$  and  $D=12$  sec)

It is clear from Tables (4.2) and (4.4) that the difference between  $S$  and  $T$  increases as  $p$  increases for all values of  $D$ . This is mainly because, as  $p$  is increases, the number of packets sent during normal periods ( $S_{NP}$ ) is drastically decreased (for example, for  $D=6$  sec,  $S_{NP}$  decreases from 12838.90 to 5.00 packets as  $p$  increases from 0.001 to 0.5. While the number of packets sent during other stages of the long delay cycle either remain unchanged or slightly change, as illustrated in Table (4.3).

Next, we compare the predicted throughput from the PLLDC model and the PFTK model against the values obtained from NS-2 simulation. A numerical comparison is presented in Table (4.4) and plotted in Figure (4.5). The results obtained for  $T$  show that the PLLDC model can predict  $T$  more accurately than the PFTK model. It is also shown that the difference between the PFTK model and the simulation result is always higher than the difference between the PLLDC model and the simulation results.

Table (4.4) Comparison of the throughput ( $T$ ) for Scenario #1										
#	$\rho$	Throughput ( $T$ )								
		PFTK model	$D$							
			6		8		10		12	
			PLLDC	NS-2	PLLDC	NS-2	PLLDC	NS-2	PLLDC	NS-2
1	0.001	130.56	109.89	120.31	104.04 (-5.3)*	108.57	98.79 (-10.1)	103.02	94.04 (-14.4)	96.35
2	0.005	52.64	46.21	48.92	43.75 (-5.3)	44.31	41.54 (-10.1)	42.39	39.54 (-14.4)	40.29
3	0.01	33.72	30.81	30.95	29.17 (-5.3)	29.72	27.69 (-10.1)	28.98	26.36 (-14.4)	26.31
4	0.05	9.393	10.05	9.85	9.52 (-5.2)	9.61	9.04 (-10.0)	9.24	8.60 (-14.4)	8.71
5	0.1	5.29	5.42	5.32	5.13 (-5.3)	5.19	4.88 (-10.0)	4.95	4.64 (-14.4)	4.92
6	0.5	0.35	0.50	-	0.47 (-6.0)	-	0.45 (-10.0)	-	0.43 (-14.0)	-

\* The values between brackets represent the percentage decrease in  $T$  as compared to the values of  $T$  when  $D=6$  sec. The negative sign indicates that  $T$  decreases as  $D$  increases.

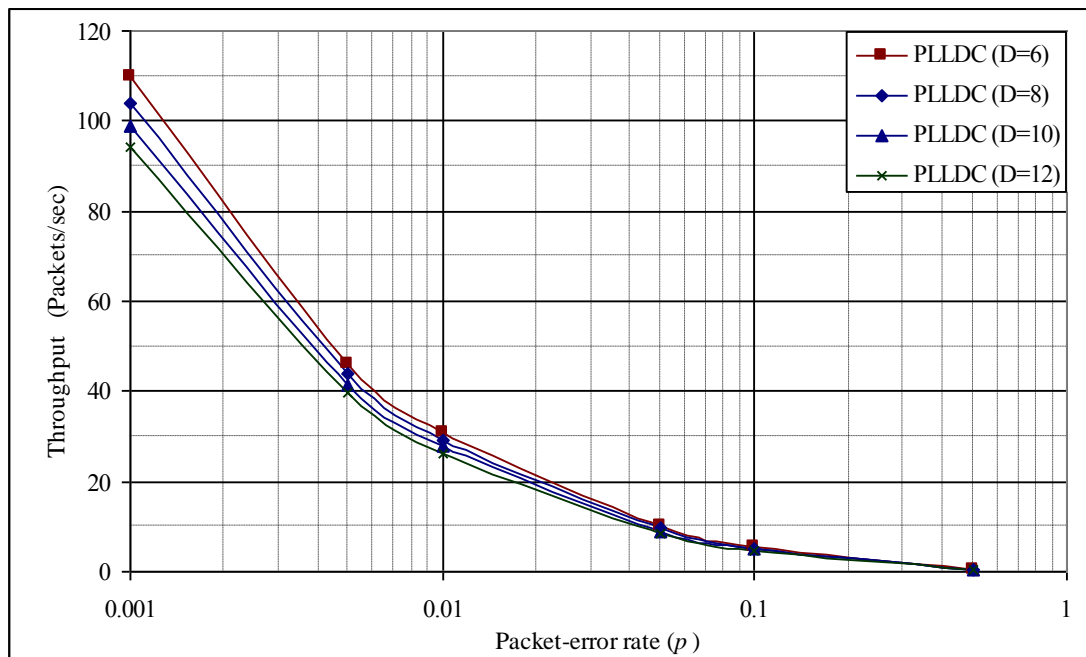


Figure (4.4). Comparison of  $T$  for Scenario #1.



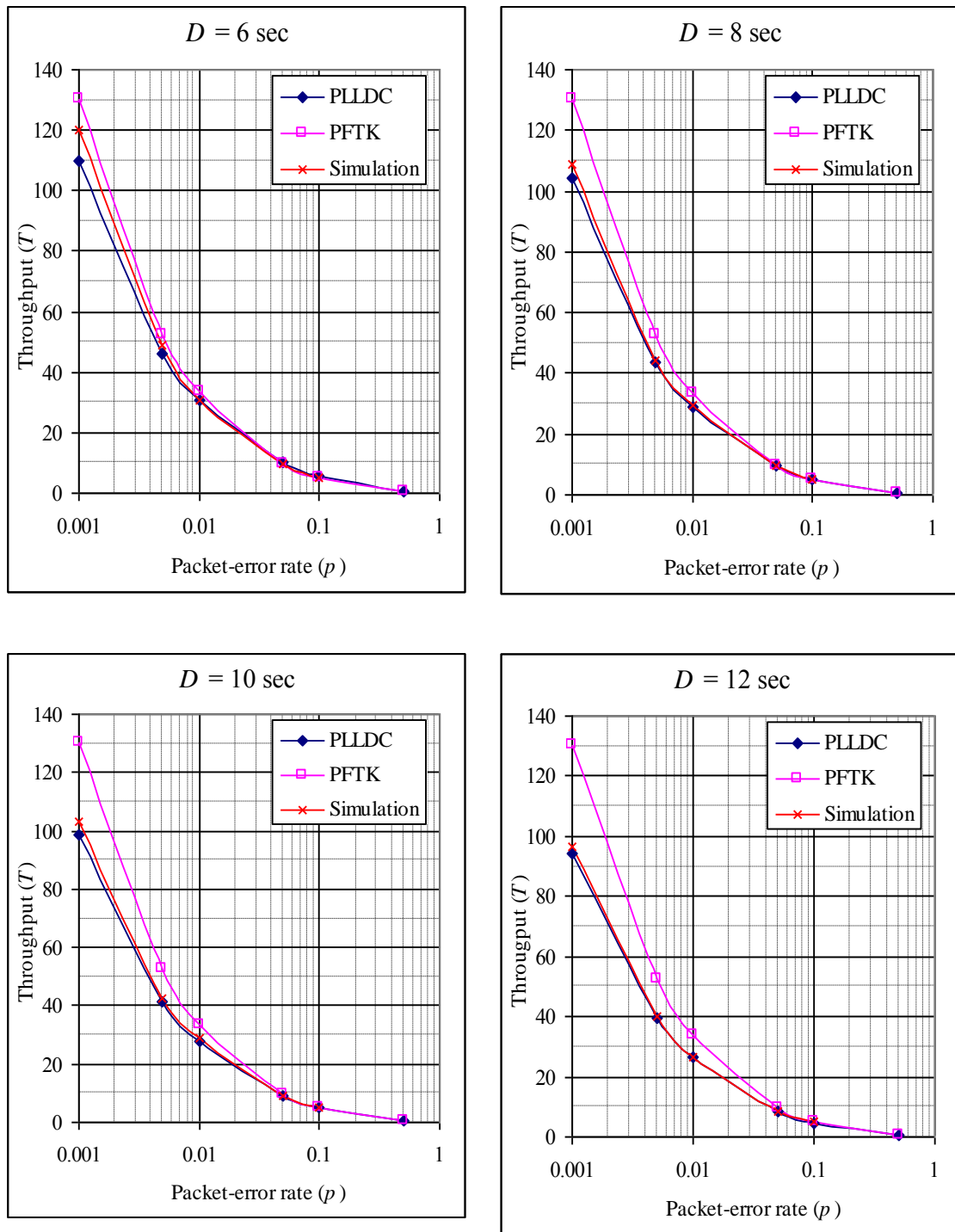


Figure (4.5). Variation of  $T$  with  $p$  for various values of  $D$  for Scenario #1.

### 4.1.3 Scenario #1: Utilization factor ( $U$ )

It is clear from the above discussions that the numeric values of  $S$  and  $T$  do not give us an apparent indication on the performance of TCP Reno in realistic wireless environment that suffers from packet-loss and long delay cycles. Therefore, in this subsection, the performance is illustrated in terms of the utilization factor, which represents the ratio between  $S$  and  $T$ . The variation of estimated  $U$  with  $p$  for various values of  $D$ , for Scenario #1, is plotted in Figure (4.6).

It clear from Figure (4.6) that  $U$  decreases as  $p$  increases due the fact that the difference between the  $S$  and  $T$  increases as  $p$  increases, which has been discussed in previous subsections. For example,  $U$  decreases from around 97% to 60% as  $p$  increases from 0.001 to 0.5. In addition, since introducing the Long Delay Cycles almost equally affects  $S$  and  $T$  for all values of  $D$ , then the values of  $U$ , and consequently the performance of TCP Reno, are unaffected by the variation in  $D$ . The results are presented in semi-logarithmic scale for clarity.

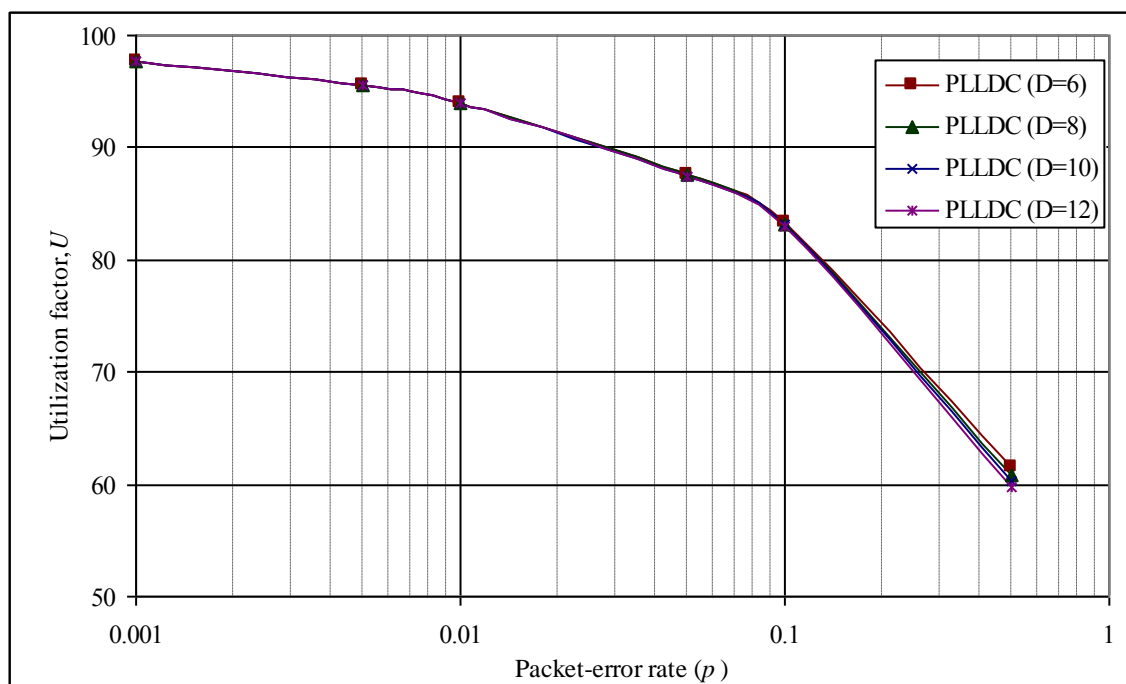


Figure (4.6). Variation of  $U$  with  $p$  for various values of  $D$  for Scenario #1.

## 4.2 Scenario #2: Investigates the Effect of Intervals between Long Delays ( $I$ )

Scenario #2 investigates the effect of  $I$  on the performance of TCP Reno. The performance is investigated by predicting the variation of  $S$ ,  $T$ , and  $U$  with  $p$  for various values of  $I$ . The list and values for the input parameters for Scenario #2 are summarized in Table (4.5).

Table (4.5) Input Parameter for Scenario #2	
Parameter	Value
Packet-loss rate ( $p$ )	0.001 to 0.5
Duration of the long delay ( $D$ )	8 sec
Interval between long delays ( $I$ )	30, 120, and 240 sec
Round Trip Time ( $RTT$ )	0.2 sec
Retransmission timeout or timeout ( $T_o$ )	1.0 sec
Slow start threshold at the end of a long delay $D$ ( $SST$ )	2
Number of packets acknowledged by one ACK packet ( $b$ )	2
Receiver's maximum congestion window size ( $W_m$ )	800 packet

### 4.2.1 Scenario #2: Sending rate ( $S$ )

The PLLDC model is used to predict the variation of  $S$  with  $p$  for three various values of  $I$  (30, 120, and 240 sec). The results obtained are listed in Table (4.6) and plotted in Figure (4.7). Figure (4.7) shows that  $S$  is inversely proportional to  $p$ , i.e., it decreases as  $p$  increases for all values of  $I$ . Furthermore, for  $p < 0.5$ ,  $S$  is directly proportional to  $I$ , while for  $p \geq 0.5$ ,  $S$  is inversely proportional. For example, as it can be seen in Table (4.6), for  $p = 0.001$ , the results demonstrate that when  $I$  increases by 4 and 8 times (from 30 to 120 and 240 sec),  $S$  increases by 18.4% (from 106.49 to 126.12 packets/sec) and 22.2% (from 126.49 to 130.09 packets/sec), respectively. While, for  $p = 0.5$ ,  $S$  decreases by 17.1% (from 0.77 to 0.64 packets/sec) and 20.7% (from 0.77 to 0.61 packets/sec), respectively. Thus, the variation of  $I$  causes either an increase or a decrease in the value of  $S$  depending on the value of the Packet-Loss rate.

Table (4.6) shows that  $S$  increases as  $l$  increases, and the estimated value of  $S$  by the PLLDC model will be equal to the value estimated by the PFTK model, when  $l$  tends to infinity (very large value). This is because as  $l$  increases the unconstructive effect of the long delays ( $D$ ) is minimized, as they are not occurring very frequently, and when  $l$  tends to infinity, the effect of  $D$  is totally eliminated, which is similar to the PFTK model. So that it is considered as an excellent way of checking the validity of the PLLDC model. However, this can be easily understood by referring to Table (4.3), which shows the number of packets sent during the different stages of a long delay cycle.

Table (4.6) Comparison of the sending rate ( $S$ ) for Scenario #2.					
#	$\rho$	Sending rate ( $S$ )			
		PFTK model	PLLDC model		
			$l=30$ sec <sup>+</sup>	$l=120$ sec	$l=240$ sec
1	0.001	134.32	106.49	126.12 (18.4)*	130.09 (22.2)
2	0.005	57.30	45.81	53.91 (17.7)	55.55 (21.3)
3	0.01	38.60	31.07	36.38 (17.1)	37.46 (20.6)
4	0.05	13.11	10.87	12.45 (14.5)	12.77 (17.5)
5	0.1	7.24	6.17	6.92 (12.2)	7.08 (14.7)
6	0.5	0.58	0.77	0.64 (-17.1)**	0.61 (-20.7)

+ Reference Value  
\* Values between brackets represent the percentage variation calculated by Eqn. (4.1).  
\*\* The negative sign indicates a decrease in the value of  $S$ .

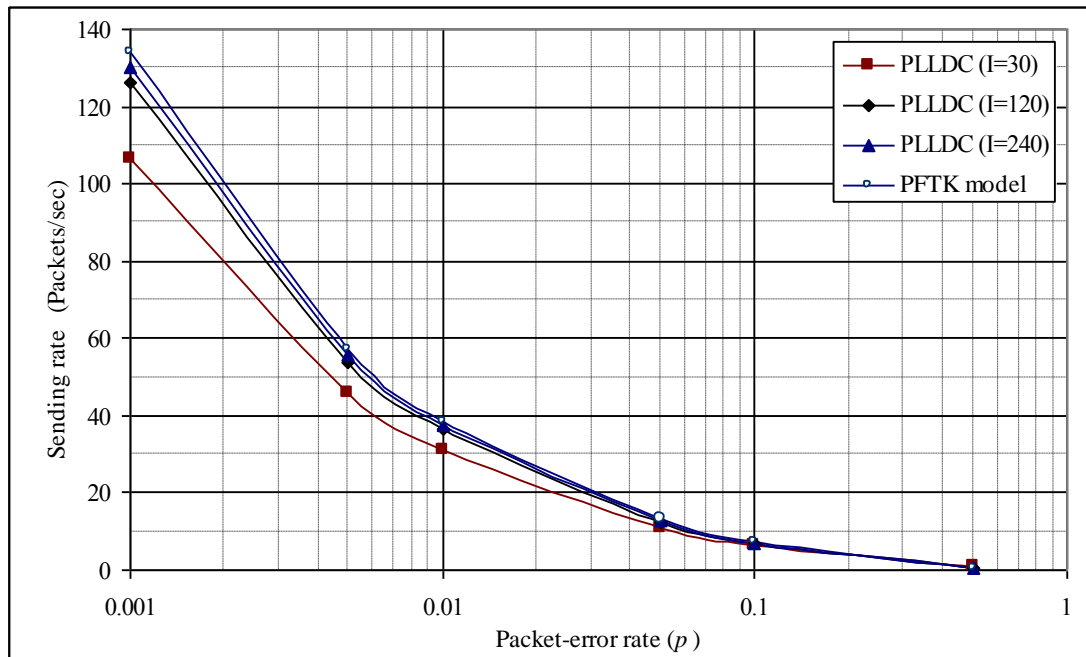


Figure (4.7). Variation of  $S$  with  $p$  for various values of  $I$  for Scenario #2.

#### 4.2.2.Scenario #2: Throughput ( $T$ )

The PLLDC model is also used to predict the variation of  $T$  with  $p$  for three various values of  $I$  (30, 120, and 240 sec). The results obtained are listed in Table (4.7) and plotted in Figure (4.8). Figure (4.8) shows that  $T$  is inversely proportional to  $p$  for all values of  $I$ .

Next, we compare the predicted throughput from the PLLDC model and the PFTK. A numerical comparison is presented in Table (4.7) and plotted in Figure (4.8). The results obtained for  $T$  show that the PLLDC model can predict  $T$  more accurately than the PFTK model. It is also shown that the difference between the PFTK model results is always higher than the PLLDC model.

It is clear from the results of  $T$  presented in Table (4.7) that  $T$  is inversely proportional to  $p$  regardless of the value of  $I$ , and, for the same value of  $p$ ,  $T$  increases as  $I$  increases, because the unconstructive effects of Long Delays ( $D$ ) are minimized as they do not occur very frequently. For the same reason, it can be seen from Table (4.7) that as  $I$  increases the value of  $T$  estimated using the

PLLDC model becomes closer to that of the PFTK model. When  $l$  tends to infinity (becomes large value), the effect of  $D$  is totally eliminated (becomes zero), which is similar to the PFTK model.

Table (4.7) Comparison of the throughput ( $T$ ) for Scenario #2					
#	$\rho$	Throughput ( $T$ )			
		PFTK model	PLLDC model		
			$l=30 \text{ sec}^+$	$l=120 \text{ sec}$	$l=240 \text{ sec}$
1	0.001	130.56	104.04	123.67 (18.9)*	127.68 (22.70)
2	0.005	52.64	43.75	51.72 (18.2)	53.33 (21.9)
3	0.01	33.72	29.17	34.33 (17.7)	35.38 (21.3)
4	0.05	9.39	9.52	10.99 (15.5)	11.29 (18.6)
5	0.1	5.29	5.13	5.82 (13.4)	5.96 (16.1)
6	0.5	0.35	0.47	0.38 (-18.1)**	0.37 (-21.8)

+ Reference Value  
\* Values between brackets represent the percentage variation calculated by Eqn. (4.1).  
\*\* The negative sign indicates a decrease in the value of  $S$ .

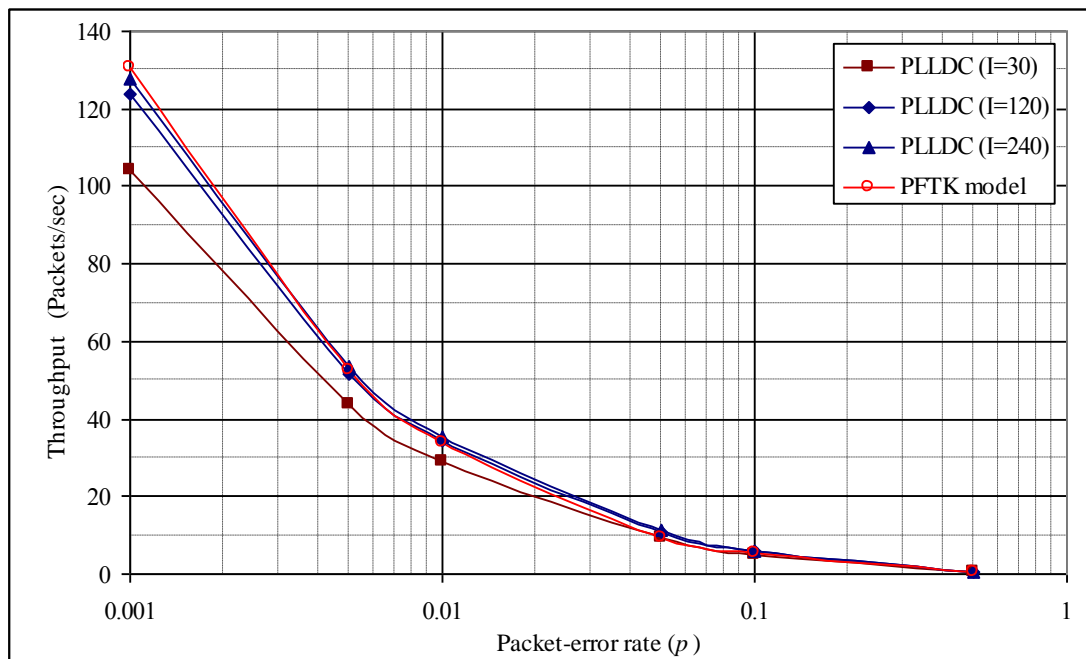


Figure (4.8). Comparison of  $T$  for Scenario #2

### 4.2.3.Scenario #2: Utilization factor ( $U$ )

In this subsection, the variation of the estimated  $U$  with  $p$  for various values of  $I$ , for Scenario #2, is presented. The results obtained are plotted in Figure (4.9).

It is clear from Figure (4.9) that  $U$  decreases as  $p$  increases due the fact that the difference between  $S$  and  $T$  increases as  $p$  increases, which has been discussed in previous subsection. For example,  $U$  decreases from around 97% to 60% as  $p$  increases from 0.001 to 0.5. In addition, since introducing the Long Delay Cycles almost equally affects  $S$  and  $T$  for all values of  $I$ , then the values of  $U$ , and consequently the performance of TCP Reno, are unaffected by the variation in  $I$ .

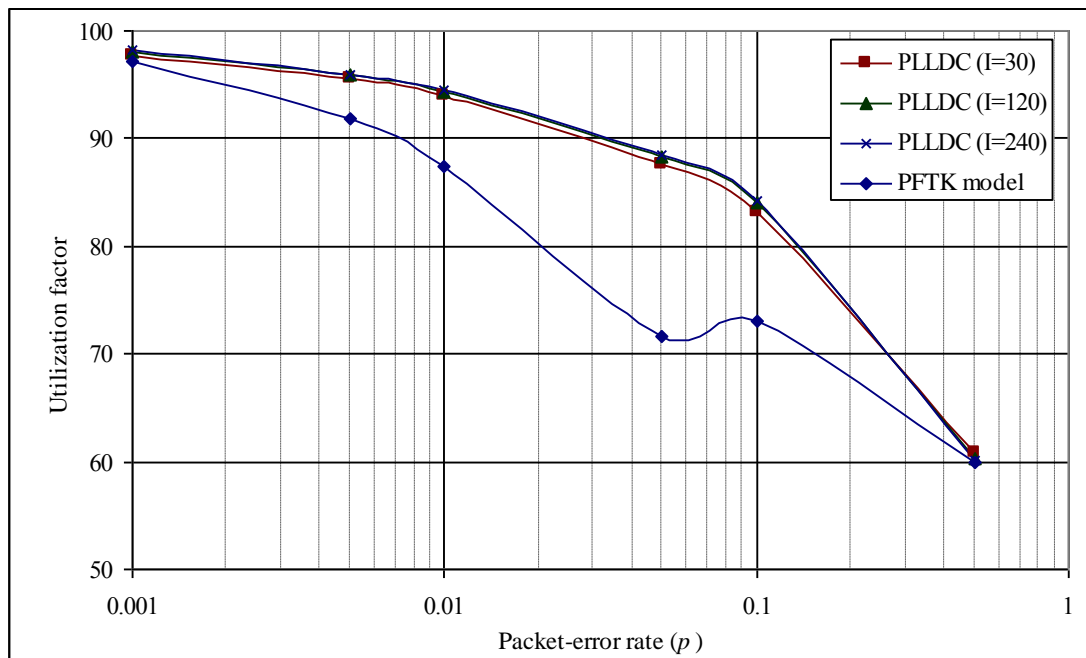


Figure (4.9). Variation of  $U$  with  $p$  for various values of  $I$  for scenario #2.

### 4.3 Scenario #3: Investigates the Effect of Round Trip Time ( $RTT$ )

Scenario #3 investigates the variation of  $S$ ,  $T$ , and  $U$  with  $p$  for various values of  $RTT$ . The list and values of input parameters are summarized in Table (4.8).

Table (4.8) Input Parameter for Scenario #3.	
Parameter	Value
Packet-loss rate ( $p$ )	0.001 to 0.5
Duration of the long delay ( $D$ )	8 sec
Interval between long delays ( $I$ )	30 sec
Round Trip Time ( $RTT$ )	0.2, 0.4, 0.6, and 0.8 sec
Retransmission timeout or timeout ( $T_o$ )	1.0 sec
Slow start threshold at the end of a long delay $D$ ( $SST$ )	2
Number of packets acknowledged by one ACK packet ( $b$ )	2
Receiver's maximum congestion window size ( $W_m$ )	800 packet

#### 4.3.1 Scenario #3: Sending rate ( $S$ )

The variation of  $S$  with  $p$  for four various values of  $RTT$  (0.2, 0.4, 0.6, and 0.8 sec) is estimated using the PLLDC and PFTK models. The results obtained are tabulated in Table (4.9) and plotted in Figure (4.10). Figure (4.10) shows that  $S$  is inversely proportional to  $p$  for all values of  $RTT$ . Furthermore, if all other network parameters remain unchanged,  $S$  decreases as  $RTT$  increases. The percentage variation of  $S$  with  $p$  with respect to reference values at  $RTT=0.2$  sec is shown in Table (4.10). The result demonstrates that a significant reduction in  $S$  occurs when  $RTT$  increases. For example, for  $p=0.001$ ,  $S$  is reduced by 75% when  $RTT$  is increased from 0.2 sec to 0.8 sec. However, the reduction in  $S$  decreases as  $p$  increases. This can be well understood by looking back to the equation for calculating  $S$  in Chapter 3, and the insight values presented in Table (4.3) in Section (4.1).

The results obtained are validated against those obtained by the PFTK model. The results obtained by the PLLDC and PFTK models are consistent, but the PFTK model always predicts higher values for  $S$  as it neglects the Long Delay,



Slow-Start stage, and the data packets sent during these periods.

Table (4.9) Comparison of the sending rate (S) between the PLLDC and PFTK models for Scenario #3.									
#	$\rho$	RTT = 0.2 sec		RTT = 0.4 sec		RTT = 0.6 sec		RTT = 0.8 sec	
		PFTK model	PLLDC model	PFTK model	PLLDC model	PFTK model	PLLDC model	PFTK model	PLLDC model
1	0.001	134.32	106.49	67.51	53.46	45.09	35.66	33.85	26.73
2	0.005	57.30	45.81	29.34	23.48	19.72	15.79	14.85	11.90
3	0.01	38.60	31.07	20.18	16.28	13.66	11.04	10.33	8.36
4	0.05	13.11	10.87	7.72	6.44	5.47	4.60	4.23	3.59
5	0.1	7.24	6.17	4.71	4.05	3.49	3.03	2.77	2.43
6	0.5	0.58	0.77	0.54	0.73	0.51	0.69	0.48	0.66

Table (4.10) Comparison of the sending rate (S) estimated by the PLLDC model for Scenario #3.					
#	$\rho$	RTT = 0.2 sec <sup>+</sup>	RTT = 0.4 sec	RTT = 0.6 sec	RTT = 0.8 sec
1	0.001	106.49	53.46 (- 49.80)*	35.66 (- 66.51)	26.73 (- 74.90)
2	0.005	45.81	23.48 (- 48.75)	15.79 (- 65.53)	11.90 (- 74.02)
3	0.01	31.07	16.28 (- 47.61)	11.04(-64.46)	8.36 (-73.08)
4	0.05	10.87	6.44 (-40.73)	4.60 (-57.70)	3.59 (-67.00)
5	0.1	6.17	4.05 (-34.32)	3.03 (-50.86)	2.43 (-60.58)
6	0.5	0.77	0.73 (-5.70)	0.69 (-10.68)	0.66 (-15.07)

+ Reference Value  
 \* Values between brackets represent the percentage variation calculated by Eqn.(4.1). The negative sign indicates a decrease in the value of  $S$ .

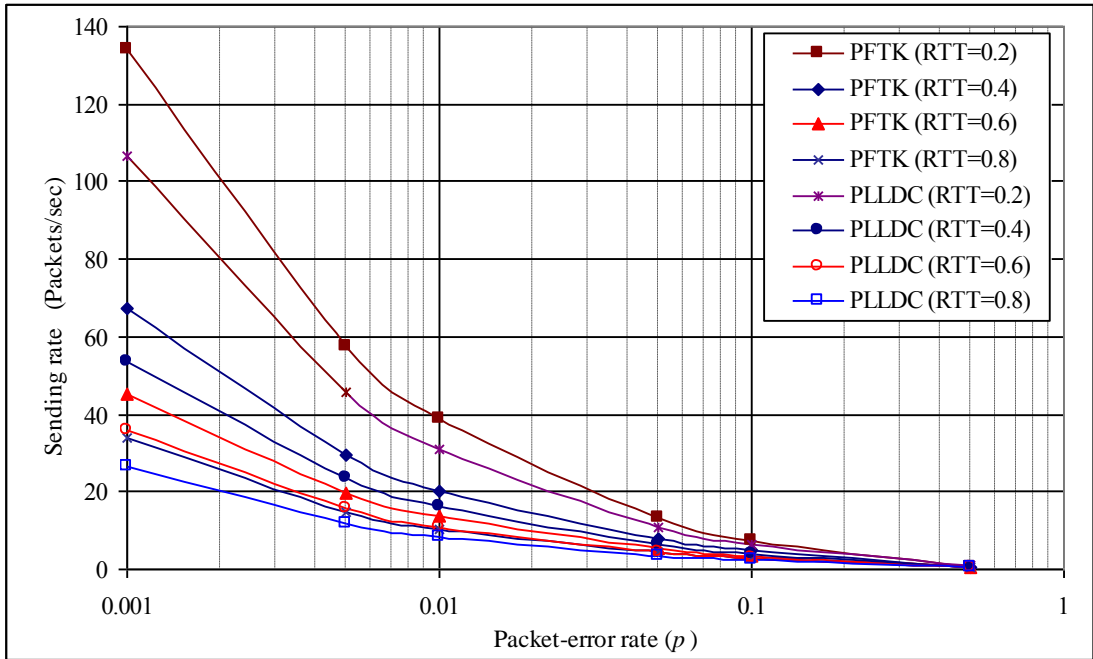


Figure (4.10). Variation of  $S$  with  $p$  for various values of  $RTT$  for Scenario #3.

#### 4.3.2 Scenario #3: Throughput ( $T$ )

The variation of  $T$  with  $p$  for four various values of  $RTT$  (0.2, 0.4, 0.6 and 0.8 sec) are computed using the PLLDC and PFTK analytical models. The results obtained are listed in Table (4.11) and plotted in Figure (4.11). For the same reasons discussed above,  $T$  decreases as  $p$  increases and also as  $RTT$  increases. The percentage decreases in  $T$  with  $RTT$  is computed using Eqn. (4.1) and listed in Table (4.12), for  $RTT$  0.4, 0.6, and 0.8, while  $T$  values at  $RTT$  0.2 sec are considered as a reference values. It is clear that the performance of TCP Reno is significantly degraded as  $RTT$  increases, and the degradation is the worst as compared to the effect of  $D$  and  $I$ . The results obtained for  $T$  show that the PLLDC model can predict  $T$  more accurately than the PFTK model. It is also shown that the difference between the PFTK model results is always higher than the PLLDC model.

Table (4.11) Comparison of the throughput ( $T$ ) for Scenario #3.									
#	$p$	$RTT = 0.2 \text{ sec}$		$RTT = 0.4 \text{ sec}$		$RTT = 0.6 \text{ sec}$		$RTT = 0.8 \text{ sec}$	
		PFTK model	PLLDC model	PFTK model	PLLDC model	PFTK model	PLLDC model	PFTK model	PLLDC model
1	0.001	130.56	104.04	65.96	51.96	44.13	34.47	33.15	25.70
2	0.005	52.64	43.75	27.56	22.29	18.66	14.90	14.11	11.16
3	0.01	33.72	29.17	18.31	15.18	12.57	10.23	9.57	7.70
4	0.05	9.39	9.52	6.04	5.60	4.45	3.96	3.53	3.07
5	0.1	5.29	5.13	3.64	3.34	2.77	2.48	2.24	1.97
6	0.5	0.35	0.47	0.33	0.44	0.31	0.42	0.29	0.40

Table (4.12) Comparison of the throughput ( $T$ ) calculated by the PLLDC model for Scenario #3.					
#	$P$	$RTT=0.2 \text{ sec}^+$	$RTT=0.4 \text{ sec}$	$RTT=0.6 \text{ sec}$	$RTT=0.8 \text{ sec}$
1	0.001	104.04	51.96 (-50.06)*	34.47 (-66.87)	25.70 (-75.30)
2	0.005	43.75	22.29 (-49.05)	14.90 (-65.94)	11.16 (-74.49)
3	0.01	29.17	15.18 (-47.94)	10.23 (-64.92)	7.70 (-73.60)
4	0.05	9.52	5.60 (-41.19)	3.96 (-58.36)	3.07 (-67.79)
5	0.1	5.13	3.34 (-34.87)	2.48 (-51.67)	1.97 (-61.56)
6	0.5	0.47	0.44 (-5.61)	0.42 (-10.51)	0.40 (-14.83)

+ Reference Value  
\* Values between brackets represent the percentage variation calculated by Eqn.(4.1). The negative sign indicates a decrease in the value of  $T$ .

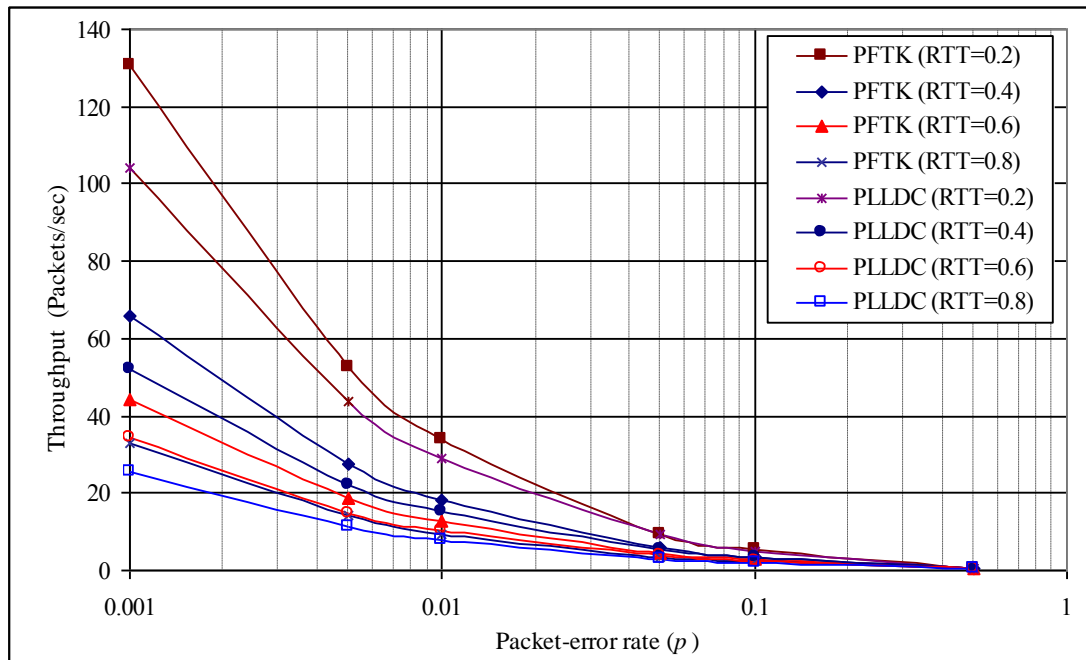


Figure (4.11). Variation of  $T$  with  $p$  for various values of  $RTT$  for scenario #3.

### 4.3.3. Scenario #3: Utilization factor ( $U$ )

The variation of the estimated  $U$  with  $p$  for various values of  $RTT$  for Scenario #3 is plotted in Figure (4.12). It is clear from Figure (4.12) that  $U$  decreases as  $p$  increases due to the fact that the difference between the  $S$  and  $T$  increases as  $p$  increases. For example,  $U$  decreases from around 97% to 60% as  $p$  increases from 0.001 to 0.5. In addition, since increasing  $RTT$  differently affects  $S$  and  $T$ , then the values of  $U$  and consequently the performance of TCP Reno gets worse as  $RTT$  increases as shown in Figure (4.12).

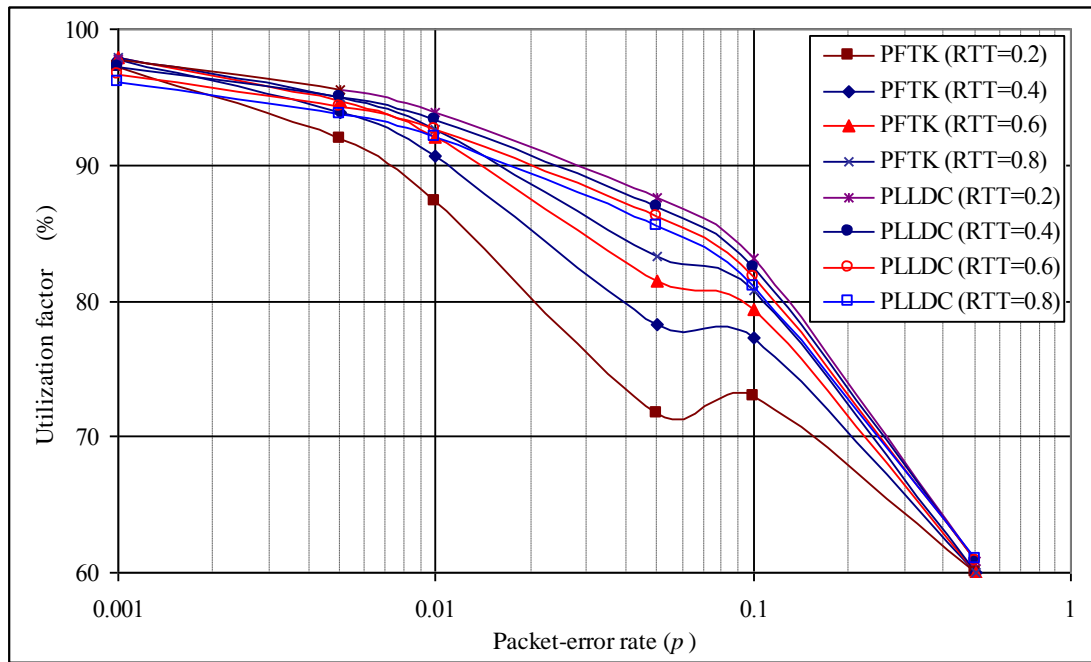


Figure (4.12). Variation of  $U$  with  $p$  for various values of  $RTT$  for scenario #3.

## Chapter 5

### Conclusions and Recommendations for Future Work

#### 5.1 Conclusions

The main conclusions of this thesis can be summarized as:

1. A new effective analytical model that can be used to evaluate the performance of TCP Reno in a realistic wireless environment that suffers from a wide-range of Packet-Loss (PL) and Long Delay Cycles (LDCs), namely, the PLLDC model is developed. The model estimates the TCP Reno Sending rate ( $S$ ), Throughput ( $T$ ), and Utilization factor ( $U$ ) as a function of environment and system-driven parameters. The former includes: Packet-Loss rate ( $p$ ), duration of the Long Delay ( $D$ ), Interval between Long Delays ( $I$ ), and Round Trip Time ( $RTT$ ), while the latter includes: Timeout ( $T_o$ ), Slow-Start Threshold at the end of a Long Delay ( $SST$ ), number of packets acknowledged by one ACK packet ( $b$ ), and the receiver's maximum congestion Window size ( $W_m$ ).
2. The new model is based on the stochastic PFTK TCP congestion control and avoidance model.
3. The accuracy of the results obtained by the PLLDC model was validated against results obtained from the well-known PFTK model and the widely-used NS-2 network simulator.
4. The effectiveness of the new analytical PLLDC model was compared with that of PFTK model, and it was found that the proposed model is much more accurate for estimating TCP performance in the presence of a wide range of Packet-Loss and frequent Long Delay Cycles (delay spikes).
5. The results obtained and the accuracy by which they were achieved demonstrate that the PLLDC model can be used effectively to evaluate the

6. performance of TCP Reno under different network environments. The performance of the TCP Reno was evaluated in terms of estimating the variation of  $S$ ,  $T$ , and  $U$  with  $p$  for various values of  $D$ ,  $I$ , and  $RTT$ .
7. A number of relationships can be deduced-out from the results obtained by performing the three scenarios that were described in Chapter 4, such as:
  - a. For Scenario #1,  $S$  and  $T$  are inversely proportional to  $D$  for all values of  $p$ . Where  $S$  and  $T$  are reduced by around 14% as  $D$  is doubled (increased from 6 sec to 12 sec) for all values of  $p$ .
  - b. For Scenario #2,  $S$  and  $T$  are directly proportional to  $I$ , where  $S$  and  $T$  increases as  $I$  increases. But, the percentage increase in  $S$  and  $T$  decreases as  $p$  increases until it is inflated when  $p \geq 0.5$ .
  - c. For Scenario #3,  $S$  and  $T$  are inversely proportional to  $RTT$  for all values of  $p$ . Where  $S$  and  $T$  are reduced by around 75% as  $RTT$  increases from 0.2 sec to 0.8 sec and  $p=0.001$ .
8. The PLLDC model can be easily modified to accommodate any variation in the operating regime of TCP Reno. Furthermore, the model can be used as an excellent teaching tool to provide an insight into the real behavior of TCP Reno in wireless networks.

## 5.2 recommendations for Future Work

The analytical model proposed in this thesis is expected to significantly contribute to a number of future studies, such as:

- iv. There is always a fundamental trade off between the rapidness of detection of true losses versus the risk of unnecessary retransmissions when designing a retransmission Timeout ( $T_o$ ) calculation algorithm or setting related parameters. For example, the minimum Retransmission Timeout, i.e., the lower bound of the RTO value and its buildup, has a significant impact on the effectiveness of the RTO estimator. There is no existing method to optimally set minimum  $T_o$ , and the current

- v. practice is to set it to twice the clock granularity. Since our proposed model considers the effect of spurious retransmission (ST), it can assist in determining an appropriate value of minimum  $T_o$ .
- vi. There is an increasing research interest to study the interaction between TCP and lower layer protocols in wireless environments. The settings of lower layer protocols, such as handoff schemes in Mobile IP and retransmission schemes at the link layer, have a non-trivial impact on the frequency of TCP spurious timeouts. The model proposed in this thesis can facilitate the fine-tuning of these settings in a more coordinated fashion in order to achieve an optimal performance.
- vii. When new modifications to TCP are made to alleviate the effects of ST, our proposed model provides a framework for evaluating the impact of the modifications, and comparing the performance of the modified TCP with previous versions of TCP. This will improve the current situation where the modifications are mainly tested by simulations, and hence may not be able to cover all possible network scenarios.
- viii. The current model estimates the TCP performance with Spurious Fast Retransmission (SFR) for an infinite receiver buffer. Therefore, it is recommended to extend the proposed model to estimate the TCP performance for finite receiver case.



## Reference

- [Abd 07] Mohamed Abdelhafez, George Riley, Robert G. Cole, and Nam Phamdo, "Modeling and Simulations of TCP MANET Worms", Georgia Institute of Technology, 2007.
- [Abo 03] Alhussein Abouzeid and Sumit Roy, "Stochastic Modeling of TCP in Networks with Abrupt Delay Variations", *Wireless Networks* Vol. 9, pp 509–524, 2003.
- [Alt 05a] Eitan Altman, Chadi Barakat, and Víctor M. Ramos R., "Analysis of AIMD Protocols over Paths with Variable Delay", *Journal of Computer Networks*, Vol. 48, Issue 6, pp. 960-971, 2005.
- [Alt 05b] Eitan Altman and Konstantin Avrachenkov, Chadi Barakat, Parijat Dube, "Performance Analysis of AIMD Mechanisms over a Multi-State Markovian Path", *Journal of Computer Networks*, Vol. 47, Issue 3, pp. 307-326, 2005.
- [Alt 05c] Eitan Altman, Konstantin Avrachenkov, C. Barakat, A. Kherani and B. Prabhu, "Analysis of MIMD Congestion Control Algorithm for High Speed Networks", *Computer Networks*, Vol. 48, No. 6, pp. 972-989, 2005.
- [Alt 00] E. Altman, K. Avrachenkov, C. Barakat, "A Stochastic Model of TCP/IP with Stationary Random Losses", *Proceedings of the ACM SIGCOMM*, 2000.
- [Arg 06a] Antonios Argyriou, "A Joint Performance Model of TCP and TFRC with Mobility Management Protocols", *Wireless Communications and Mobile Computing*, Vol. 6, pp. 547–557, 2006.
- [Arg 06b] Antonios Argyriou, Vijay Madiseti, "Modeling the Effect of Mobile Handoffs on TCP and TFRC Throughput", *GLOBECOM*, 2006.

- [Bac 02] F. Baccelli, D. R. McDonald, and J. Reynier, "A Mean-Field Model for Multiple TCP Connections Through a Buffer Implementing RED", *Journal of Performance Evaluation*, Vol. 49, Issues 1-4, pp. 77-97, 2002.
- [Bla 04] Ethan Blanton and Mark Allman, "Using TCP DSACKs and SCTP Duplicate TSNs to Detect Spurious Retransmissions ", RFC 3708, 2004.
- [Bud 04] A. Budhiraja, F. Hernández-Campos, V.G. Kulkarni, and F. D. Smith, "Stochastic Differential Equation for TCP Window Size: Analysis and Experimental Validation", in *Probability in the Engineering and Informational Sciences*, Vol. 18, pp. 111– 140. 2004.
- [Che 08] Jiwei Chen, Mario Gerla, Yeng Zhong Lee, and M.Y. Sanadidi, "TCP with Delayed ACK for Wireless Networks", *Journal of Ad Hoc Networks*, Vol. 6, Issue 7, pp. 1098-1116, 2008.
- [Chr 05] Nicolas Christin and Jörg Liebeherr, "Marking Algorithms for Service Differentiation of TCP Traffic", *Journal of Computer Communications*, Vol. 28, Issue 18, pp. 2058-2069, 2005.
- [Com 06] Douglas E. Comer, **Internetworking with TCP/IP: Principles Protocols, and Architecture**, 5<sup>th</sup> edition, Prentice-Hall, 2006.
- [Dun 06] Roman Dunaytsev, Yevgeni Koucheryavy, and Jarmo Harju, "The PFTK-Model Revised", *Journal of Computer Communications*, Volume 29, Issues 13-14, pp. 2671-2679, 2006.
- [Eom 02] Seop Eom, HeyungSub Lee, and Masash Suganoet, Masayuki. Murata and Hideo. Miyahara, "Improving TCP Handoff Performance in Mobile IP Based Networks", *Computer Communications*, Vol. 25, No. 7, pp. 635–646, 2002.

- [Flo 03] S. Floyd, "High speed TCP for Large Congestion Window", RFC 3649, December 2003.
- [For 07] Behrouz A. Forouzan, **Data Communications and Networking**, 4<sup>th</sup> edition, McGraw-Hill, 2007.
- [Fu 03] Shaojian Fu and Mohammed Atiquzzaman, "Modeling TCP Reno with Spurious Timeout in Wireless Mobile Environment", International conference on computer communication and Network, Dallas, 2003.
- [Gur 02] Andrei Gurtov and Reiner Ludwig, "Making TCP Robust against Delay Spikes", Internet Draft, draft-gurtov-tsvwg-tcp-delay-spikes-00.txt, 2002.
- [Gur 01a] Andrei Gurtov, "Effect of Delays on TCP Performance", in International Federation for Information Processing (IFIP) Personal Wireless Communications, 2001.
- [Gur 01b] Andrei Gurtov, "Making TCP Robust Against Delay Spikes", University of Helsinki, Department of Computer Science, Technical Report, No.C-2001-53, 2001.
- [Has 03] Mahbub Hassan and Raj Jain, **High Performance TCP/IP Networking: Concepts, Issues, and Solutions**, Prentice-Hall, 2003.
- [Hes 05] Joao P. Hespanha, "A Model for Stochastic Hybrid Systems with Application to Communication Networks", Journal of Nonlinear Analysis, Vol. 62, Issue 8, pp. 1353-1383, 2005.
- [Ho 08] Cheng-Yuan Ho, Yaw-Chung Chen, Yi-Cheng Chan, and Cheng-Yun Ho, "Fast Retransmit and Fast Recovery Schemes of Transport Protocols: A Survey and Taxonomy", Journal of Computer Networks, Vol. 52, Issue 6, pp. 1308-1327, 2008.

- [Hou 08] Christos N. Houmkozis and George A. Rovithakis, "A Neuro-Adaptive Congestion Control Scheme for Round Trip Regulation", Journal of Automatica, Vol. 44, Issue 5, pp. 1402-1410, 2008.
- [Kam 03] Ahmed E. Kamal, "Discrete-Time Modeling of TCP Reno under Background Traffic Interference with Extension to RED-Based Routers", Journal of Performance Evaluation, Vol. 58, Issues 2-3, pp. 109-142, 2004.
- [Kes 05a] Alexander Kesselman and Yishay Mansour, "Optimizing TCP Retransmission Timeout", Proceedings of the 4th International Conference on Networking (ICN'05), Vol. 2, pp. 133-140, 2005.
- [Kes 05b] Alexander Kesselman and Yishay Mansour, "Adaptive AIMD Congestion Control", Special Issue on Network Design, Vol. 43, No. 1-2, pp. 97-111, 2005.
- [Kha 02] F. Khafizov and M. Yavuz, "Running TCP over IS-2000", in IEEE International Conference on Communications, New York, pp.3444–3448, 2002.
- [Kim 07] S.P. Kim and K. Mitchell, "An Analytic Model of TCP Performance over Multi-hop Wireless Links with Correlated Channel Fading", Journal of Performance Evaluation, Vol. 64, Issue 6, pp. 573-590, 2007.
- [Kli 08] Dzmitry Kliazovich, Fabrizio Granelli, and Daniele Miorandi, "Logarithmic Window Increase for TCP Westwood+ for Improvement in High Speed, Long Distance Networks", Journal of Computer Networks, Vol. 52, Issue 12, pp. 2395-2410, 2008.
- [Kug 03] H.T Kung, Koan-Sin Tan and Pai-Hsiang. Hsiao, "TCP with Sender-Based Delay Control", Computer Communications, Vol. 26, No. 14, pp. 1614-1621, 2003.

- [Kuu 00] P. Kuusela, P. Lassila, J. Virtamo, and P. Key, "Modeling RED with Idealized TCP Sources", 9<sup>th</sup> IFIP Conference on Performance Modelling and Evaluation of ATM & IP Networks, 2001.
- [Lai 02] Yuan-Cheng Lai and Chang-Li Yao, "Performance Comparison between TCP Reno and TCP Vegas", Journal of Computer Communications, Vol. 25, Issue 18, pp. 1765-1773, 2002.
- [Lee 08] Myungjin Lee, Moonsoo Kang, Myungchul Kim, and Jeonghoon Mo, "A Cross-Layer Approach for TCP Optimization over Wireless and Mobile.
- [Lei 05] D. J. Leith, P. Clifford, "Using the 802.11e EDCF to Achieve TCP Upload Fairness over WLAN Links", WiOpt, April 2005.
- [Les 07] Marios Lestas, Andreas Pitsillides, Petros Ioannou, and George Hadjipollas, "Adaptive Congestion Protocol: A Congestion Control Protocol with Learning Capability", Journal of Computer Networks, Vol. 51, Issue 13, pp. 3773-3798, 2007.
- [Li 01] Kang Li, Molly Shor, Jonathan Walpole, Calton Pu, and David Steere, "Modeling the Effect of Short-term Rate Variations on TCP-Friendly Congestion Control Behavior", American Control Conference 2001.
- [Lud 05] Reiner Ludwig, Andrei Gurtov, "The Eifel Response Algorithm for TCP", RFC 4015, 2005.
- [Lud 01] Reiner Ludwig, "The TCP Retransmit (rxt) flag", Internet Draft, draft-ludwigsvwg- tcp-rxt-flag-02.txt, 2001.
- [Lud 00] Reiner Ludwig, Randy H. Katz, "The Eifel Algorithm: Making TCP Robust Against Spurious Retransmissions", Appears in ACM Computer Communications Review, Vol. 30, No. 1, 2000.

- [Lul 04] M. Lulling and J. Vaughan, "A Simulation-Based Performance Evaluation of Tahoe, Reno and Sack TCP as Appropriate Transport Protocols for SIP", Journal of Computer Communications, Vol. 27, Issue 16, pp. 1585-1593, 2004.
- [Mal 08] David Malone, Douglas J. Leith, Anshuman Aggarwal and Ian Dangerfield, "Spurious TCP Timeouts in 802.11 Networks", Workshop on Wireless Network Measurement (WiNMee 2008). 2008.
- [Mal 06] Sireen Malik and Ulrich Killat, "Lighter and Faster Simulations of High-Speed IP Networks", AEU - International Journal of Electronics and Communications, Vol. 60, Issue 7, pp. 494-503, 2006.
- [Mas 06] Saverio Mascolo, "Modeling the Internet Congestion Control Using a Smith Controller with Input Shaping", Journal of Control Engineering Practice, Vol. 14, Issue 4, pp. 425-435, 2006.
- [Mol 03] Niels Moller and Karl Henrik Johansson, "Influence of Power Control and Link-Level Retransmissions on Wireless TCP", Computer Science Vol. 2811, 2003.
- [Ng 05] A. Ng, D. Malone, and D. Leith, "Experimental Evaluation of TCP Performance and Fairness in an 802.11e Test-Bed", Proceeding of ACM SIGCOMM Workshops, 2005.
- [Pad 00] Jitendra Padhye, Victor Firoiu, Don Towsley, and Jim Kurose, "Modeling TCP Reno Performance: a Simple Model and its Empirical Validation", IEEE/ACM Transactions on Networking, Vol. 8, No. 2, pp. 133-145, 2000.
- [Pap 07] Panagiotis Papadimitriou and Vassilis Tsaoussidis, "On TCP Performance over Asymmetric Satellite Links with Real-Time Constraints", Journal of Computer Communications, Vol. 30, Issue 7, pp. 1451-1465, 2007.

- [Sta 08] Williams Stallings, Business Data Communications, 6th Edition, Prentice-Hall, 2008. "Networks", Journal of Computer Communications, Vol. 31, Issue 11, pp. 2669-2675, 2008.
- [Tan 03] Andrew Tanenbaum, **Computer Networks**, 4<sup>th</sup> edition, Prentice Hall, 2003.
- [Tek 08] Mohamed Tekala and Robert Szabo, "Dynamic Adjustment of Scalable TCP Congestion Control Parameters", Journal of Computer Communications, Vol. 31, Issue 10, pp. 1890-1900, 2008.
- [Ven 03] Andrea De Vendictis, Andrea Baiocchi, and Michela Bonacci, "Analysis and Enhancement of TCP Vegas Congestion Control in a Mixed TCP Vegas and TCP Reno Network Scenario", Journal of Performance Evaluation, Vol. 53, Issues 3-4, pp. 225-253, 2003.
- [Voi 07] Laura M. Voicu, Steven Bassi, and Miguel A. Labrador, "Analytical and Experimental Evaluation of TCP with an Additive Increase Smooth Decrease (AISD) Strategy", Journal of Computer Communications, Vol. 30, Issue 2, pp. 479-495, 2007.
- [Wei 05] Michele C. Weigle, Kevin Jeffay, and F. Donelson Smith, "Delay-Based Early Congestion Detection and Adaptation in TCP: Impact on Web Performance", Journal of Computer Communications, Vol. 28, Issue 8, pp. 837-850, 2005.
- [Wid 01] Joerg Widmer, Robert Denda, and Martin Mauve, "A Survey on TCP-friendly congestion control", Special Issue of the IEEE Network Magazine Control of Best Effort Traffic, Vol. 15, No. 3, pp. 28-37, 2001.
- [Xin 06] Fei Xin and Abbas Jamalipour, "TCP Performance in Wireless Networks with Delay Spike and Different Initial Congestion Window Sizes", Journal of Computer Communications, Vol. 29, Issue 8, pp. 926-933, 2006.

- [Yan 03] Y. Richard Yang, Min Sik Kim, and Simon S. Lam, "Transient Behaviors of TCP-Friendly Congestion Control Protocols", Journal of Computer Networks, Vol. 41, Issue 2, pp. 193-210, 2003.
- [Yav 02] M. Yavuz and F. Khafizov, "TCP Over Wireless Links with Variable Bandwidth", 56th IEEE Vehicular Technology Conference, Vancouver, pp.1322–1327, 2002.



