TCP PERFORMANCE ENHANCEMENT OVER WIRELESS NETWORKS

by

Aiyathurai Jayananthan

A thesis submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Canterbury New Zealand

2007

Approved by

Chairperson of Supervisory Committee

Program Authorized
to Offer Degree

Date

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

UNIVERSITY OF CANTERBURY

ABSTRACT

TCP PERFORMANCE IMPROVEMENT OVER WIRELESS NETWORKS

by Aiyathurai Jayananthan

Chairperson of the Supervisory Committee: Professor Harsha Sirisena
Department of Electrical and Computer Engineering

Transmission Control Protocol (TCP) is the dominant transport protocol in the Internet and supports many of the most popular Internet applications, such as the World Wide Web (WWW), file transfer and e-mail. TCP congestion control algorithms dynamically learn the network bandwidth and delay characteristics of a network and adapt its performance to changes in traffic so as to avoid network collapse.

TCP is designed to perform well in traditional wireline networks with the assumptions that packet losses are mainly due to network congestion and random bit error rate (BER) is negligible. However, networks with wireless links suffer from significant packet losses due to random bit errors and handoffs. Hence TCP performs poorly in networks with wireless links because it treats any packet loss in the network to be a result of network congestion and slows down its transmission rate, or even cause the TCP sender to experience unnecessary timeouts, further reducing its performance.

The development of advance wireless networks, such as WiFi, UMTS and WiMAX, make it necessary to find ways to improve TCP's efficiency and resource utilization, as well as improve the user's experience and reduce latency times. In order to find effective solutions to this effect, packet losses across wireless links should be distinguished from congestion related packet losses.

In this thesis, we concentrate on two main strategies for enabling the TCP congestion control mechanism to determine the cause for a packet loss. One is a proxy-based mechanism that monitors the radio network interface and sends radio network feedback (RNF) to the TCP sender with the status of the wireless link. The other one is an end-to-end mechanism, in which the packet error pattern is used as the system metric to fine-tune the congestion control

mechanism. It also presents an analytical model of TCP with enhanced recovery mechanism for wireless environments.

In a proxy-based mechanism, TCP sender is explicitly informed of any effects caused by wireless links. However, the implementation technique is network dependent. We have proposed and developed three proxy-based schemes; the radio network feedback (RNF) scheme over an 802.11 WLAN network, the radio network controller (RNC) feedback over a UMTS network and a wireless enhancement proxy (WENP) over both the 802.11 WLAN and UMTS networks.

The RNF scheme is introduced at the 802.11 WLAN base station that monitors the TCP packet flows over the wireless links, detects wireless packet losses and provides feedback to the TCP sender using one of the TCP header reserved control bits, called RNF flag. TCP Reno is modified to utilize the radio network feedback to distinguish the losses due to wireless effects form the congestion and fine-tuned to perform wireless enhanced fast retransmit and fast recovery mechanisms. The RNF scheme is implemented using the OPNET tool, and the simulation results show that the TCP performance is significantly improved.

The RNC feedback mechanism, similar to the RNF scheme, is developed and implemented in a UMTS network. The GPRS Tunneling Protocol (GTP) layer of the UMTS Radio Network Control (RNC) protocol stack was modified to detect and notify the TCP sender of the wireless packet losses, which is the main difference between the RNF and RNC mechanisms. The simulation results shows that the RNC feedback mechanism significantly improves the TCP performance compared to that of standard TCP over UMTS.

The wireless enhancement proxy (WENP) is developed to minimize spurious TCP timeouts over wireless networks and implemented in both 802.11 WLAN and UMTS networks. WENP extends the proposed RNF and RNC feedback mechanisms to detect both wireless packet losses and large delays across the wireless link, and to notify the TCP sender of these events with the aid of two reserved bits in the TCP header. TCP Reno is further modified to utilize the WENP feedback to distinguish both wireless packet losses from congestion losses and spurious timeouts from normal timeouts. It is also fine-tuned to perform both the wireless enhanced fast retransmit and fast recovery mechanism and the timeout mechanism. The simulation results demonstrate that the proposed scheme markedly improves the TCP performance compared to that of standard WLAN and UMTS implementations.

An end-to-end early packet loss recovery (EPLR) mechanism that modifies the TCP Reno fast retransmit algorithm to detect packet losses early and to speed up the packet recovery process to reduce the number of TCP timeouts over networks with heavy packet losses, such as wireless networks is also presented. TCP Reno with EPLR scheme is implemented in a UMTS network and its performance is compared with that of TCP Reno and New Reno. Simulation results shows that Reno with EPLR improves the TCP performance and application response time significantly compared to that of both Reno and New Reno by reducing the TCP timeouts, which is the main cause of degradation of the TCP performance in a wireless environment.

Finally, we develop an analytical TCP throughput model with enhanced TCP Reno fast retransmit algorithm to avoid timeouts. The model captures the TCP fast retransmit mechanism and expresses the steady state congestion window and throughput as a function of network utilization factor, round trip time (RTT) and loss rate. Another new feature added to the model is dynamic adjustment of the congestion window size depending on the packet drop rates. This speeds up the packet recovery process and reduces the number of TCP timeouts over networks with heavy packet losses. The proposed model is implemented over a UMTS network and its performance is compared with that of TCP Reno. Simulation results show that the proposed model reduces the TCP timeouts and improves the TCP performance compared to that of TCP Reno. It is also found that the model provides a very good match to the steady-state congestion window behavior.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF EQUATIONS

ACKNOWLEDGMENTS

I would like to express my deep and sincere gratitude to my research supervisors, Professor Dr. Harsha Sirisena and Professor Dr. Krzysztof Pawlikowski for their guidance and encouragement to carry out this research endeavor. Their brilliant ideas and suggestions helped me to improve the presentation of this thesis.

My very special thanks go to Professor Dr. Harsha Sirisena for his valuable advice and friendly help. His extensive discussions and constructive comments have been very helpful for this study.

Many thanks to Professor Dr. Krzysztof Pawlikowski for allowing me to use Computer Science facilities during the first year of my Ph.D study.

I am also grateful for the Electrical and Computer Engineering Department for providing me an excellent work environment during the past years and for computer staff, Pieter Kikstra and Florin Predan, for their cheerful assistance.

Many thanks to all members of Networking Research Group of Electrical and Computer Engineering, and Computer Science Departments for their discussion during weekly meetings.

I also want to express my acknowledgements to University of Canterbury for providing me financial support for this work.

I am also thankful to the Royal Society of New Zealand for providing me travel grant to attend the ATNAC 2006 conference in Melbourne.

I feel a deep sense of gratitude for my late father and mother who formed part of my vision and taught me the good things that really matter in life. The happy memory of my father still provides a persistent inspiration for my journey in this life.

Lastly, and most importantly, I wish to thank my wife Pathmaverni, for her love, patience and support during my Ph.D study. I am also thankful to my lovely sons, Nirojan and Vanujan, who provided me with an additional joyful dimension to our life mission.

GLOSSARY

AP	Access Point
ACK	Acknowledgement
A-TCP	Adaptive TCP
AIMD	Additive-Increase, Multiplicative-Decrease
ARP	Address Resolution Protocol
ARPA	Advanced Research Projects Agency
ARPANET	Advanced Research Projects Agency Network
AIRMAIL	Asymmetric Reliable Mobile Access in Link-layer
ATM	Asynchronous Transfer Mode
ARQ	Automatic Repeat Request
BW	Bandwidth
BS	Base Station
BS	Base Station
BSS	Basic Service Set
BER	Bit Error Rate
BCH	Broadcast Channel
BCCH	Broadcast Control Channel
BMC	Broadcast/Multicast Control
CSMA/CA	Carrier Sense Multiple Access with Collision Avoidance
CS	Circuit-Switched
CTS	Clear To Send
CDMA	Code Division Multiple Access
CDM	Code Division Multiplexing

CCCH	Common Control Channel
CPCH	Common Packet Channel
CA	Congestion Avoidance
CE	Congestion Experienced
CWND	Congestion Window
CWR	Congestion Window Reduction
CID	Connection ID
CFP	Contention Free Period
CP	Contention Period
CCH	Control Channel
CN	Core Network
CRC	Cyclic Redundancy Check
DCH	Dedicated channel
DCCH	Dedicated Control Channel
DoD	Department of Defense
DS	Direct Sequence
DCF	Distributed Coordination Function
DIFS	Distributed Inter Frame Space
DS	Distribution System
DSCH	Downlink Shared Channel
DUPACK	Duplicate Acknowledgement
EPLR	Early Packet Loss Recovery
ETD	Early Timeout Detection
ECT	ECN Capable Transport
ECE	ECN-Echo (ECE)
ETSI	European Telecommunications Standards Institute

ECN	Explicit Congestion Notification
ELN	Explicit Loss Notification
ELNR	Explicit Loss Notification to Receiver
ELN-ACK	Explicit Loss Notification with Acknowledgement
EWMA	Exponential Weighted Moving Average
ESS	Extended Service Set
FR	Fast Retransmit
FTP	File Transfer Protocol
FH	Fix Host
FS	Flight Size
FACH	Forward Access Channel
FACK	Forward Acknowledgment
FEC	Forward Error Control
FEC	Forward Error Correction
FER	Forward Error Correction
FER	Frame Error Rate
FDD	Frequency Division Duplex
FDMA	Frequency Division Multiple Access
FH	Frequency Hopping
GGSN	Gateway GPRS Support Node
GPRS	General Packet Radio Service
GPRS	General Packet Radio Service
GSM	Global System for Mobile Communications
GBN	Go-Back-N
GMM	GPRS Mobility Management
GSN	GPRS Support Node

GTP	GPRS Tunneling Protocol
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
I-TCP	Indirect TCP
IEEE	Institute of Electronics and Electrical Engineers
IFS	Inter Frame Space
ISO	International Standard Organization
ITU	International Telecommunication Union
IMT-2000	International Telecommunications-2000
ICMP	Internet Control Message Protocol
IP	Internet Protocol
LAN	Local Area Network
MSS	Maximum Segment Size
MTU	Maximum Transfer Unit
MAC	Medium Access Control
MH	Mobile Host
MSC	Mobile Switching Center
METP	Mobile-End Transport Protocol
MSR	Mobility Support Routers
NSF	National Science Foundation
NACK	Negative Acknowledgement
NAV	Network Allocation Vector
OSI	Open System Interconnection
OFDM	Orthogonal Frequency Division Multiplexing
OVSF	Orthogonal Variable Spreading Factor
PDCP	Packet Data Convergence Protocol

PDN	Packet Data Network
PDP	Packet Data Protocol
PS	Packet-Switched
PCH	Paging Channel
PCCH	Paging Control Channel
PCF	Point Coordination Function
PIFS	Point Inter Frame Space
PSTN	Public Switched Telephone Network
QoS	Quality of Service
RF	Radio Frequency
RLC	Radio Link Control
RLP	Radio Link Protocol
RNC	Radio Network Control
RNC-FB	Radio Network Control Feedback
RNF	Radio Network Feedback
RNF	Radio Network Feedback
RNS	Radio Network Subsystems
RRC	Radio Resource Control
RRM	Radio Resource Management
RACH	Random Access Channel
RED	Random Early Detection
RTS	Ready To Send
RcvWND	Receiver Window
RFC	Request for Comment
RAND	research and development
RTO	Retransmit Timeout

RTT	Round Trip Time
SACK	Selective Acknowledgement
SRP	Selective Repeat Protocol
SR	Selective Retransmissions
SAP	Service Access Points
SGSN	Serving GPRS Support Node
SRNC	Serving Radio Network Controller
SIFS	Short Inter Frame Space
SMART	Simple Method to Aid Retransmission
SS	Slow-Start
SSTHRESH	Slow-Start Threshold
SMG	Special Mobile Group
SCMTP	Split-Connection Mobile Transport Protocol
SF	Spreading Factor
SYN	Synchronization
3G	Third Generation
3GPP	Third Generation Partnership Project
TDD	Time Division Duplex
TDM	Time Division Multiplexing
TDMA	Time-Division Multiple Access
TCH	Traffic Channel
TCP	Transmission Control Protocol
TPC	Transmission Power Control
TULIP	Transport Unaware Link Improvement Protocol
UMTS	Universal Mobile Telecommunications System
UTRAN	UMTS Terrestrial Radio Access Network

UDP	User Datagram Protocol
UE	User Equipment
VoD	Video-on-Demand
VLR	Visitor Location Register
W-CDMA	Wideband Code Division Multiple Access
WiMAX	WiMAX
WEP	Wired Equivalent Privacy
WENP	Wireless Enhancement Proxy
WLD	Wireless Loss Detector
WLN	Wireless Loss Notification
W-RTO	Wireless Retransmit Timeout
W-RTT	Wireless Round Trip Time
WTD	Wireless Timeout Detection
WTN	Wireless Timeout Notification
WTCP	Wireless Transmission Control Protocol
ZWA	Zero Window Advertisement
NSFNET	National Science Foundation Net
WLAN	Wireless Local Area Network

Chapter I.

Introduction

The Internet has revolutionized the computer and communications world and developed into the world's largest information network. It is a source of news, facts and figures and has become an essential part of modern civilization. The convergence of Internet, telephony and wireless technologies, such as WiFI, UMTS and WiMAX, changes the way we communicate, work and live. The present challenge of leading telecommunications and networking vendors is to provide systems with richer functionality at faster speeds and lower cost in order to meet constantly evolving market demands.

Computer networks should be well designed and optimized to get maximum benefit with minimal cost. Most wire-line networks are optimized to perform well under different network conditions. However, TCP applications in mobile and wireless networks experience severe performance degradation because packet losses due to bit errors and handoffs initiate congestion control mechanism. This leads to an absolute necessity to design and optimize the TCP congestion control mechanism to effectively handle the non-congestion related issues in wireless environments. This dissertation presents some novel approaches to the design of protocols and enhancement proxies for TCP congestion control mechanism.

1.1 THE EVOLUTION OF TCP/IP AND THE INTERENET

The Advanced Research Projects Agency (ARPA) was created by US in response to the launch of Sputnik, first artificial earth satellite, by Soviet Union in 1957. ARPA had the mission of advancing science and technology applicable to the military [TANENBAUM. A.S, 3rd ed]. The existing traditional circuit-switched telephone networks were considered to be too vulnerable since the loss of one line or switch would disable the entire network. The research and development (RAND) corporation came up with the idea of building a network without a central point of control. In this way, the system would not be vulnerable to a direct hit on a single location.

To accommodate this requirement, ARPA decided to adapt to a packet-switched network, consisting of a subnet and host computers. In 1969, a group of people working for the ARPA linked computers at UCLA, Standard Research Institute, the University of Utah, and the University of California at Santa Barbara to create the network. The non-centralized network was born and dubbed ARPANET (Advanced Research Projects Agency Network).

Further experiments demonstrated that the existing ARPANET protocol were not suitable for running multiple networks. Thus the ability to connect to multiple networks together in a seamless way became one of the major design goals. This led to more research on protocols, culminating with the invention of the TCP/IP model and protocols [Cerf & Kahn, 1974]. A later perspective was given to the TCP/IP model in [Leiner, Cole, Postel, & Mills, 1985].

In 1980, the Department of Defense (DoD) mandated TCP/IP protocol as an official network standard. The number of networks, machines, and users connected to the ARPANET grew rapidly after TCP/IP became the only official protocol on January 1, 1983. The National Science Foundation (NSF) chose TCP/IP when it built a nationwide research network in 1985. The collection of interconnected TCP/IP networks such as ARPANET, NSFNET and private networks became the prototype of the Internet that eventually grows to the today's global network [Leiner, Cole, Postel, & Mills, 1985].

1.2 OPEN SYSTEM INTERCONNECTION REFERENCE MODEL

The International Standard Organization (ISO) proposed the Open System Interconnection (OSI) reference model [Zimmermann, 1980] [Tanenbaum, 2003], shown in Figure 1.1, for the standardization of computer network protocols. The OSI reference model is composed of seven layers, each specifying particular network functions, and provides a conceptual framework for communication between computers.

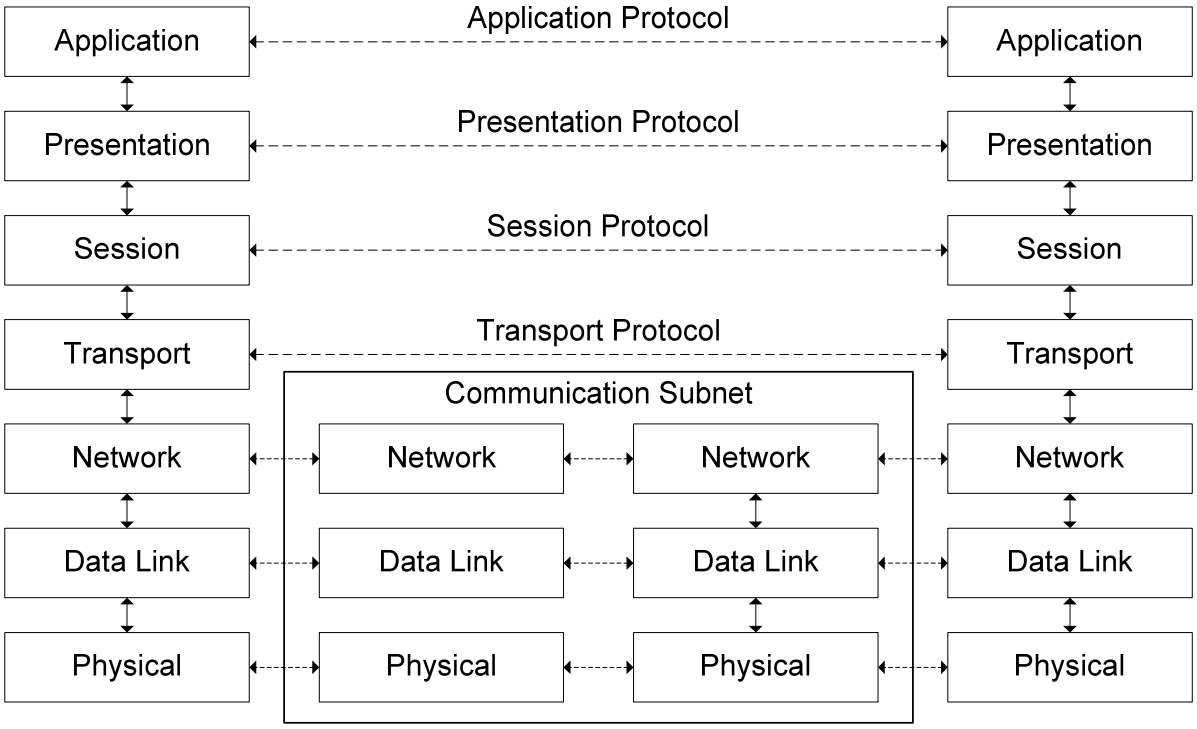

Figure 1-1 The OSI reference model

Actual communication is achieved by using communication protocols; a formal set of rules and conventions that govern how computer exchange information over network medium. One OSI layer communicates with another layer to make use of the services provided by that layer. The services provided by adjacent layers help a given OSI layer to communicate with its peer layer in other computer systems. The OSI protocols have not become as prevalent as one may expect, given the degree to which OSI has been predicted as the basis for networking. The protocol suite that has attained a stronger foothold is TCP/IP.

1.3 INTERNET GROWTH

Until 1995, the usage of the Internet was limited to file transfer, remote access to computers, and simple mail transfer in the form of a file transfer [Jamalipour, 2003]. Invention of Hypertext Transfer Protocol (HTTP) and Hypertext Markup Language (HTML) has revolutionized the Internet as the new media for telecommunications. The web browsing is considered the main factor in the popularity of the Internet and a huge increase in the Internet subscription happened after the invention of web browsers such as Netscape and Internet Explorer. Figure 1.2 shows the growth in the number of Internet users over the last decade. With the extensive progress

achieved during the last decade in wireless access technology, the wireless Internet will be the next revolutionizing factor in the Internet growth.

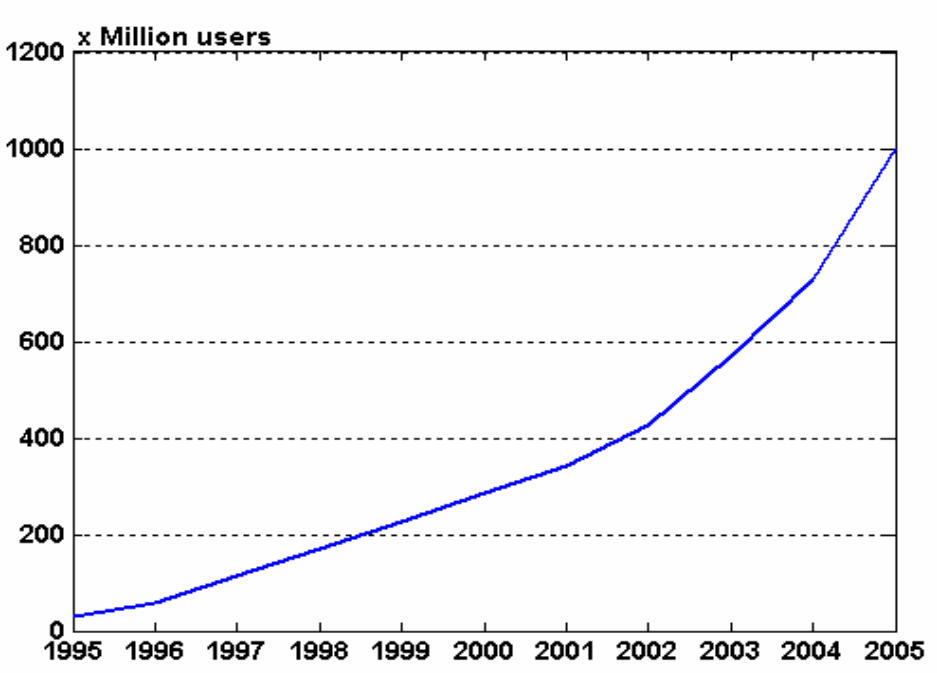

Figure 1-2 The growth in the number of Internet usage

1.4 TRENDS TOWARDS WIRELESS INTERNET

The convergence of Internet, telephony and wireless technology changes the way we communicate, work and live. Wireless communications have become pervasive. The number of mobile phones and wireless Internet has increased significantly in recent years. As shown in Figure 1.3, the number of worldwide mobile subscribers increases exponentially with no sign of a stop or a slowing down of the increase rate, while the increase in the number of fixed subscribers has been very smooth since 2002 [Jamalipour, 2003; Mohr & Konhauser, 2000].

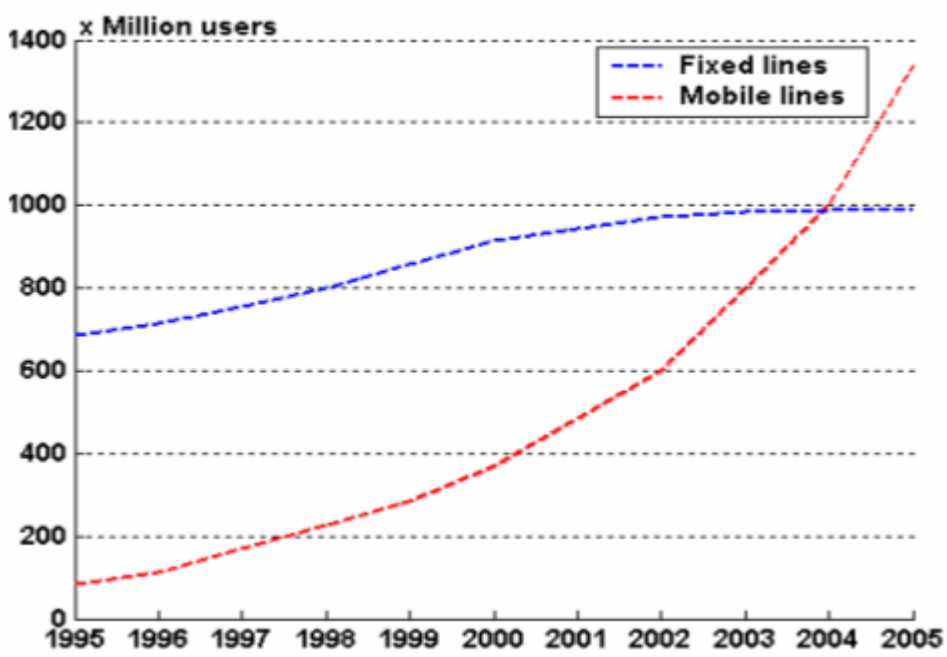

Figure 1-3 Increase in user population in fixed and mobile communications system

Mobile communications are determined by economic and technical trends and, in future, by application requirements. With the evolution of second-generation systems and the emerging third-generation systems, more advanced data and multimedia services are becoming available in addition to the mobile telephony. These trends and requirements are affecting the vision of future systems beyond the third generation [Mohr & Konhauser, 2000].

1.5 PROBLEM STATEMENT

Congestion control is the problem of managing network traffic or a network state where the total demand for resources such as bandwidth among the competing users exceeds the available capacity. It is a core infrastructural problem stemming from the packet switched and statistically multiplexed nature of the Internet and has an impact on the Internet stability and manageability.

Although TCP is the most common transport protocol used in the Internet for years, it has been shown that its congestion control algorithm lacks the ability to adapt to the wireless environments. TCP is primarily designed for wired networks, where data is seldom lost or corrupted due to link errors and queue overflow in routers is the predominant reason for the packet loss. In a wireless network, however, packet losses will occur more often due to high Bit Error Rates (BERs) than due to congestion. When using TCP over wireless networks, it

considers each packet loss as a sign of congestion and invokes congestion control measures at the source. This results in severe performance degradation.

It is highly undesirable for a protocol to react to random losses the same way as it reacts to congestion indications. TCP should react to congestion rather than packet losses. Due to the characteristics of the air interface, wireless links could introduce sporadic packet losses due to burst of packet losses. A loss of a single packet often has a little effect on network applications, but multiple packet losses can have a significant effect. TCP's inability to distinguish a loss due to congestion from a random loss can lead to serious performance degradation.

Moreover, TCP can yield low throughput in highly mobile environments due to hand-offs, which may introduce temporal disconnections, buffer losses and increased latency. Shadowing and fading of the radio signal may also cause the destination to be temporarily unavailable, which causes the TCP either to time out or to stop the transmission. The lack of mechanism to notify the TCP of this effect introduces extra delay and increases the application response time considerably.

In the case of large-scale mobility, the third Generation (3G) cellular networks are the most suitable candidates for support of Internet traffic, since they offer capacity for enhanced broadband data transfers, as well as improved transmission quality. In Code Division Multiple Access (CDMA), soft hand-off, where a mobile is connected to more than one Base Station (BS), can eliminate temporal disconnections. However, this problem may still occur if soft hand-off is not initiated promptly. TCP performances in 3G CDMA networks are generally degraded by increased latency due to the extensive processing required at the physical layer of these links for coding and interleaving, and to link layer processing for Forward Error Control (FEC) and link-level retransmission. Moreover, dynamic resource sharing among all the users in a particular cell introduces significant bandwidth variations to which TCP is unable to adapt.

In summary, TCP is very sensitive to packet losses and requires further improvements to better adapt to the wireless environments. It should be able to distinguish wireless related losses from the congestion related losses and be fine tuned to utilize the available network resources efficiently.

1.6 FOCUS OF THIS THESIS

The aim of this thesis is to improve the TCP performance over wireless networks such as WLAN and UMTS. It studies an extensive literature on the performance of TCP and emphasizes the ways to distinguish the effects due to congestion losses from the effects due to wireless errors. It addresses the TCP congestion avoidance and control issues over the wireless links from both the End-to-End and proxy based methods by developing an analytical model and Radio Network Feedback (RNF) mechanism, thereby leading to efficient network resource utilization and improving the application response time. It also analyzes widely used TCP End-to-End congestion control algorithm and presents a spectrum of new algorithms that enables the TCP to better adapt to the wireless environments.

The newly developed analytical model, RNF mechanism and algorithms are tested by performing simulations experiments using a wide range of simulation scenarios. Since it is very difficult to cover the entire area of congestion avoidance and control issues and due to the time constraint, our research scope is restricted to the improvement of TCP performances over wireless networks by avoiding the hand-offs effects.

1.7 SOLUTION APPROACH

The basic idea is to explicitly inform the TCP source of any effects caused by non-congestion related packet losses. It can be achieved by monitoring the radio interface, which requires a proxy, and by considering the packet loss rates or packet loss patterns at the TCP source. The introduction of a proxy and its implementation technique is network dependent. This thesis considers the following significant contributions to achieve the object:

- Development and implementation of the RNF technique in a WLAN Server and in a WLAN Router together with the network utilization factor (Chapter 4).

- Development and implementation of Radio Network Controll (RNC) feedback mechanism in a UMTS network (Chapter 5).

- Development and implementation of Wireless Timeout Detection (WTD) in a WLAN and in a UMTS network (Chapter 6).

- Development and design of improved congestion avoidance and control algorithm for Early Timeout Detection (ETD) (Chapter 7).

- Development and design of an analytical model of TCP with enhanced recovery mechanism for wireless environments (Chapter 8).

1.8 LIMITATIONS

The proposed solutions could not implement in a CDMA network because the simulation tool, OPNET, used in this study does not support the CDMA network. The RNF and RNC feedback mechanism do not consider the hand-off effects due to mobility. However, they can be further extended to support mobility by introducing appropriate modifications. In addition to these, the following assumptions are made in this thesis:

- IP datagram is not encrypted so that the RNF proxy will be able to monitor the TCP flows.
- Traffic consists of File Transfer Protocol (FTP) over TCP
- Packet losses are uniformly distributed
- Receiver window is bigger than the congestion window and hence it does not influence the sender rate
- Congestion control schemes are window based and not rate based

1.9 STRUCTURE OF THESIS

This thesis comprises nine chapters and the remainder of this dissertation is organized as follows.

Chapter 2 gives an overview of all major types of TCP and briefly explains their specific functionalities. The strength and weaknesses of those TCP flavors are discussed and the TCP Reno is selected for further development. It then gives an in-depth analysis of its congestion avoidance and control mechanism and directs attention to specific areas, where further improvements required optimizing the TCP performance over wireless medium.

Chapter 3 presents an up-to-date survey of the schemes proposed to alleviate the poor End-to-End TCP performance in wireless medium. It summarizes these protocols and points out the

advantages and disadvantages of each scheme. It then briefly outlines the proposed schemes and their advantages over previously proposed schemes.

Chapter 4 gives an overview of WLAN technology and motivates the need for the RNF mechanism. The concept behind the RNF mechanism and the methodology applied in developing the RNF mechanism are explained. It then outlines the implementation details of the RNF mechanism in a WLAN environment and compares its performance with the standard WLAN and the WLAN with Snoop enhancing proxy. It also presents the guidelines for further improvement.

Chapter 5 gives an overview of UMTS technology and, based on the RNF mechanism implemented in WLAN environments, devises the RNC feedback mechanism. It provides with an incentive to the RNC development and outlines the advantages of the RNC mechanism over the RNF. The RNC mechanism is developed and implemented in a UMTS network. The TCP performance with the RNC proxy is analyzed and compared with that of the standard UMTS.

Chapter 6 analysis the adverse effect on the network performances due to the spurious TCP timeouts and motivates the development of WTD scheme. Both the RNF and RNC mechanisms are extended with WTD scheme and are implemented in a WLAN and UMTS network respectively. The TCP performance over the WLAN and UMTS networks, with and without the WTD schemes, are explained and compared.

Chapter 7 gives an in-depth analysis of the existing TCP Reno congestion avoidance and control mechanism and indicates its inability to deal with situations where it cannot initiate the congestion control, thereby leading to unnecessary timeouts. Based on the analysis, it develops the ETD scheme with improved congestion avoidance and control algorithm that enables the TCP source to early detect timeouts and to act accordingly. The ETD scheme is implemented in a UMTS network and its performance is compared with that of the standard TCP Reno.

Chapter 8 develops an analytical model of TCP by extending the work done in Chapter 7. It also proposes a further modification that dynamically adjusts its congestion window by considering the packet loss rate as the input parameter. The model is implemented in a UMTS network and its performance is explained and compared with that of TCP Reno. The guidelines for further improvement are also presented.

Finally, Chapter 9 presents the overall conclusions and indicates future directions of research.

1.10 PUBLICATION LIST

The papers prepared during this study are listed below:

[1] A. Jayananthan and Harsha Sirisena, 'TCP Performance Enhancement over WLAN with Wireless Loss Detection Proxy', *Proceedings of 5th IEEE International Conference on ICICS 2005*, pp. 654-658, Bangkok, December, 2005.

[2] A. Jayananthan, Harsha Sirisena and Krzysztof Pawlikowski, 'TCP Performance Enhancement over UMTS Network with RNC Feedback', *AusWireless'06*, Sydney, March 2006.

[3] A. Jayananthan, Harsha Sirisena and Krzysztof Pawlikowski, 'Improving TCP Performance over 802.11 WLAN with Radio Network Feedback', Australian Telecommunication Network and Applications Conference (ATNAC 2006), Melbourne, December 2006.

[4] A. Jayananthan and Harsha Sirisena, 'TCP End-to-End Performance Improvement over Wireless Networks via Early Packet Loss Recovery', Australian Telecommunication Network and Applications Conference (ATNAC 2006), Melbourne, December 2006.

[5] A. Jayananthan, Harsha Sirisena and Vijay Garg, 'Analytical Model of TCP with Enhanced Recovery Mechanism for Wireless Environments', IEEE International Conference on Communications 2007, ICC-2007, Glasgow, June 2007.

[6] A. Jayananthan, Harsha Sirisena and Krzysztof Pawlikowski, '[1] TCP Enhancement over Wireless Links by Minimizing Spurious TCP Timeouts', Australian Telecommunication Network and Applications Conference (ATNAC 2007), Christchurch, December 2007.

Chapter II.

Transmission Control Protocol

It is essential to be thoroughly familiar with TCP to understand the historic, current and future architecture of the Internet protocols. Most applications on the Internet use TCP because its built in reliability and flow control ensure safe delivery of data across an unreliable IP layer below. IP alone is a basic datagram service and does not support any concept of a session or connection. Once a datagram is sent or received, the service retains no memory of the entity with which it was communicating. The abilities to retransmit data or check it for errors are minimal or nonexistent in the datagram services.

This chapter provides a brief overview of TCP development and its general features. All major types of TCP and their specific functionalities, with special emphasis on their congestion avoidance and control mechanisms are presented. The strength and weaknesses of those TCP flavors are discussed and most appropriate transport protocol, which can be further developed to perform well in wireless networks, is selected. Finally, an in-depth analysis of the congestion avoidance and control mechanism of the selected TCP is given and the specific areas, where further improvements required optimizing its performance over wireless medium, are indicated.

2.1 DEVELOPMENT OF TCP

TCP is both complex and evolving transport protocol. The basic functionality of TCP is defined in [RFC 793] and was published in 1981. Since then, significant enhancements have been made and proposed. Host Requirements for Internet Hosts [RFC 1122] clarifies a number of TCP protocol implementation requirements. TCP extensions have been defined in by [RFC 1323], [RFC 2018] and [RFC 2481]. TCP congestion Control [RFC 2581], one of the most important TCP related Request for Comment (RFC) in recent years, describes updated congestion control algorithms to avoid congestion.

Congestion occurs when the demand is greater than the available resources, such as bandwidths of links, buffer space and processing capacity at the intermediate nodes such as

routers. Congestion control is concerned with allocating the resources in a network such that network can operate at an acceptable performance level when the demand exceeds the capacity of the network resources. Careful design is required to provide good service under heavy load. Otherwise, there can be a congestion collapse that is highly resource wasteful and causes undesirable state of operation.

Congestion collapse was first observed during the early growth phase of the Internet in the mid 1980s [RFC 896]. It was mainly due to TCP connections unnecessarily retransmitting packets that were either in transit or had already been received at the receiver. The original TCP implementations [RFC 793] used window-based flow control to control the use of buffer space at the receiver and Go-Back-N retransmission after a packet drop for reliable delivery, but did not include dynamic adjustment of the flow-control window in response to congestion.

Different types of congestion collapse are categorized in [Fall & Floyd, 1996]: *classical congestion collapse*, which occurs when the network is flooded with unnecessary retransmitted packets [Nagle, 1984] and was fixed with modern TCP retransmit timer and congestion control algorithm [Jacobson, 1988], *fragmentation-based congestion collapse* , which is given in [Kent & Mogul, 1987] and was fixed with Maximum Transfer Unit (MTU) discovery [RFC 1063, 1988], and *congestion collapse from undelivered packets*, which occurs when networks overloaded with packets that are discarded before they reach the receiver [S. Floyd & Fall, 1999].

The popularity of the Internet has caused a proliferation in the number of TCP implementations. Some of these may fail due to logic errors, or misinterpretations of the specification [RFC 2525]. Others may deliberately be implemented with the congestion control algorithms that use the available resources more aggressive than other TCP implementations. The consequence of such applications may lead to a state where effectively no congestion control and the Internet is chronically congested [RFC 2309, 1998]. There is also a significant number of TCP non-compatible and non-responsive bandwidth hungry traffic flows in the Internet, which can also pose significant threats to the stability of the Internet.

The development of TCP must avoid making radical changes that may stress the deployed network into congestion collapse, and also must avoid a congestion control *arms race* among competing protocols [RFC 2914]. TCP has experienced number of changes in its primitive design, during its development process, over the last three decades. The exponential growth in

the Internet usage increased the congestion problems. Consequently, many versions of TCP exist today. Presently, all major types of TCP employs congestion control algorithms, which include *slow-start* (SS), congestion avoidance and fast retransmit and fast recovery.

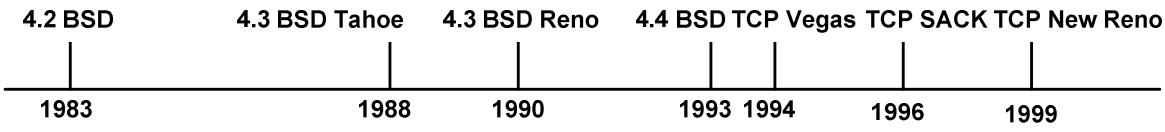

Figure 2-1 Time line for important types of TCP

2.2 GENERAL FEATURES IN TCP

TCP is an end-to-end, point-to-point transport protocol used in the Internet. Being point-to-point protocol means that there is always a single sender and a single receiver for a TCP session. Being an end-to-end protocol, on the other hand, means that TCP session should cover all parameters and transportations involved from the source host to the destination host [Jamalipour, 2003]. TCP provides connection-oriented, reliable byte stream service. We, in turn, discuss the meaning for each of these descriptive terms.

2.2.1 CONNECTION- ORIENTED

Before any data transfer could be started, a connection must be established through a process called three-way handshake. During this process, the TCP sender and receiver come to an agreement in the establishment of a connection and set the relevant parameters such as Maximum Segment Size (MSS). For example, if a client computer is contacting a server to send it some information, a TCP connection is established by exchanging control messages as follows:

- The client sends a packet with the *SYN* bit set and a sequence number *N*.
- The server then sends a packet with an *ACK* number of *N+1*, the *SYN* bit set and a sequence number *X*.
- The client sends a packet with an *ACK* number *X+1* and the connection is established.

Such signaling period before the exchange of data could sometimes put an unacceptable delay in the applications that are sensitive to delay such as real-time voice.

2.2.2 RELIABILITY

A number of mechanisms, namely checksums, duplicate data detection, sequencing, retransmissions and timers, help TCP to provide reliable data delivery. All TCP segments carry a checksum, which is used by the receiver to detect corrupted data. TCP keeps track of bytes received in order to detect and drop duplicate transmissions. In packet switched network, packets can arrive out of sequence. TCP delivers the byte stream data to an application in order by properly sequencing segments it receives. Corrupted or lost data must be retransmitted in order to guarantee delivery of data. The use of positive acknowledgements by the receiver to the sender confirms successful reception of data. The lack of positive acknowledgements, coupled with a timeout period, calls for a retransmission. TCP maintains a collection of static and dynamic timers on data sent. The TCP sender waits for the receiver to reply with an acknowledgement within a bounded length of time. If the timer expires before receiving any acknowledgement, the sender can retransmit the segment.

2.2.3 BYTE STREAM DELIVERY

TCP interfaces between the application layer above and the network layer below. A stream of 8-bit bytes is exchanged across the TCP connection between the two applications. An application sends data to TCP in 8-bit byte streams, which is then broken by TCP sender into segments in order to transmit data in manageable pieces to the receiver. The size of the application layer payload is variable but may not be larger than MSS, which is usually announced by the TCP receiver during connection establishment using the MSS option in the TCP header. However, it is limited by the outbound link's Maximum Transfer Unit (MTU). Alternatively, the sender may use the path MTU discovery [RFC 1191] to derive an appropriate MSS.

2.3 TCP SEGMENT FORMAT

The TCP segment consists of a TCP header followed by a payload. The payload includes information data passed from the application layer above for transmission. The TCP header includes address information for the segment and all information required for implementation of algorithms used in TCP. An option field is included in the TCP header that can include specific information for a particular TCP connection. The default TCP header size is 20 bytes. However, this may go up to 60 bytes with inclusion of an option field. To this effect, a header length filed is also included in the TCP header as shown in Figure 2.2.

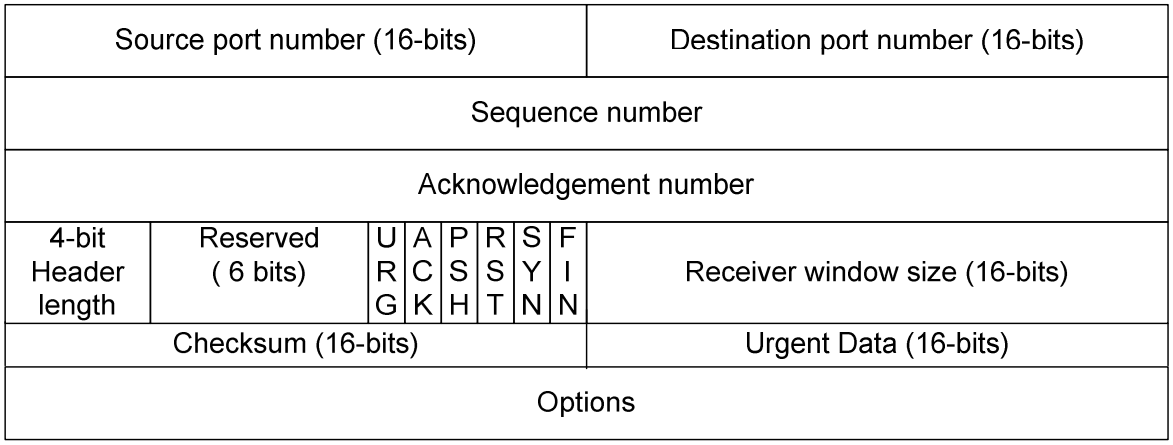

Figure 2-2 TCP segment header format

TCP segments are sent as IP datagram. The IP header carries several fields, including the source and destination host addresses. A TCP header follows the IP header, supplying information specific to the TCP protocol. This division allows for the existence of host level protocols other than TCP.

2.4 TCP FLOW CONTROL

TCP flow control is provided through the well-known sliding window mechanism. ACKs sent by the TCP receiver carry the advertised window, which limits the number of bytes the TCP sender may have outstanding at any time. The advertised window corresponds to the size of TCP receiver's receive socket buffer. The key feature of the sliding window protocol is that it permits pipelined communication to better utilize the channel capacity. The sender can send a maximum W frames without acknowledgement, where W is the window size of the sliding window. The sliding window maps to the frames in sender's buffer that are to be sent, or have been sent and now are waiting for acknowledgement. For maximum throughput, the amount of data in transit at any given time should be the channel bandwidth-delay product, which refers to the product of a data link's capacity (in bits per second) and its end-to-end delay (in seconds).

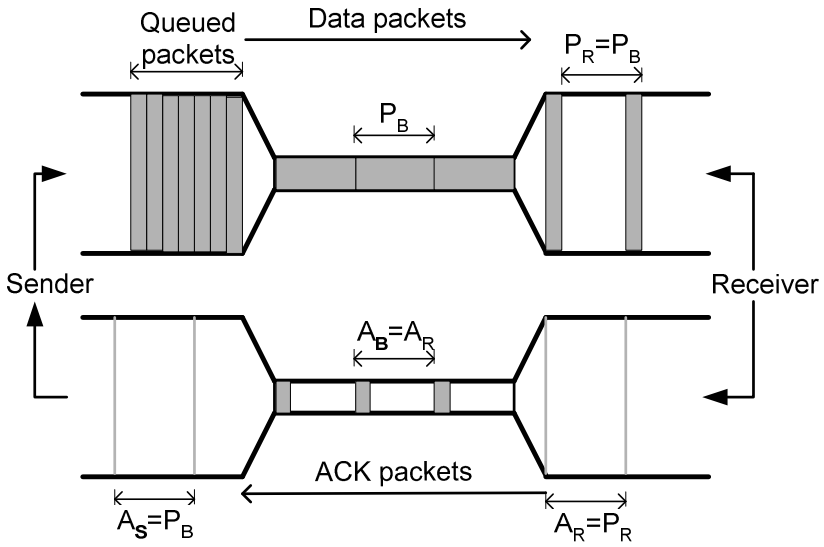

Figure 2-3 Window flow control 'self-clocking'

End-to-end protocols that implement sliding window flow control, like TCP, share an important self-clocking property. A schematic representation of a sender and receiver on high bandwidth networks connected by a slow link, the bottleneck link, is shown in Figure 2.3 [Jacobson, 1988]. The vertical and horizontal dimensions are the bandwidth (BW) and time respectively. Each of the shaded boxes represents a packet. The area of each box is the packet size because 'BW-delay product = bits'. The number of bits in a packet does not change as it goes through the network so a packet on the slow link has to spread out more in time.

Figure 2.3 shows the ideal case in which a single sender fully utilizes the non-shared bottleneck link, the slowest link in the path, with a fixed bandwidth and always sends fixed size segments. In this case, the ACK inter-arrival time (A_S) at the sender is constant and equal to the packet transmission delay over the bottleneck link, P_B. This constant stream of returning ACKs is referred to as ACK clock. The arrival of an ACK moves the sliding window to the right by one segment and clocks out a new segment, thereby keeping the number of outstanding packets, i.e. the window constant.

2.5 TCP TIME-OUT MECHANISM

In order to avoid long delays when there is no response from the receiver in a TCP connection, a time-out mechanism is employed. Therefore, after each TCP segment transmission by a sender, a timer is set and it starts counting down. If the TCP sender does not receive a

threshold number of ACKs before the timer expires, it assumes that either the packet or the ACK is lost, and retransmits the same packet again until an ACK is received. The TCP retransmit timeout (RTO) value must be carefully chosen. If RTO value is too small, the time expires quickly and premature time-outs will be generated during the usual TCP operation and thus unnecessary retransmission will occur. On the other hand, if RTO value is too large, the TCP will slowly respond to the segment loss, which means longer end-to-end delay and can also degrade performance. Therefore, the RTO value must be optimized to the extent possible.

When a packet is sent over a TCP connection, the sender times how long it takes for it to be acknowledged, producing a sequence of round-trip samples. Older TCP implementations only time one segment per RTT, whereas newer implementations use the timestamp option [RFC 1323] to time every segment. Timing every segment allows much closer tracking of changes in RTT. We refer to the RTT sampling rate as the number of RTT samples the TCP sender captures per RTT divided by the TCP sender's load. In case the TCP sender times every segment and the TCP receiver acknowledges every segment, the RTT sampling rate is 1. If the TCP sender times every segment and the TCP receiver acknowledges every other segment (delayed-ACK), the RTT sampling rate is 0.5. The closer the sampling rate to 1 the more accurately the TCP sender measures the RTT.

TCP uses a mechanism to estimate the round-trip time (RTT) in the network, based on which the timer can be set accordingly. This will be done continually so that a variable estimation will happen. TCP collects information on the most recent RTTs and then makes an average value, called a sample RTT [Kurose & Ross, 2005]. The EstimatedRTT is then computed in an iterative manner by using the following equation:

$$\text{EstimatedRTT} = (1-\alpha)\ \text{EstimatedRTT} + \alpha\ \text{SampleRTT} \qquad \text{Equation 2.1}$$

Where α is a constant between 0 and 1 that control how rapidly the estimated RTT adapts to changes and the typical value for α is 0.125 [Jacobson, 1988], which decides trade-off between efficiency and fairness. This method is called exponential weighted moving average (EWMA) owing to the inclusion of the factor α. The method provides that the influence of given sample decreases exponentially fast and puts more weight on the recent sample instead.

In addition to having an estimate of the RTT, it is also valuable to have a measure of the variability of the RTT. [RFC 2988] defines the RTT variations, DevRTT, as an estimate of how much SampleRTT typically deviates from EstimatedRTT:

$$DevRTT = (1 - \beta)\, DevRTT + \beta\, |SampleRTT - Estimated\ RTT| \qquad \text{Equation 2.2}$$

Note that DevRTT is an EWMA of the difference between SampleRTT and EstimatedRTT. If the SampleRTT values have little fluctuation, then DevRTT will be small; on the other hand, if there is a lot of fluctuation, DevRTT will be large. The recommended value of β is 0.25 [Jacobson, 1988].

After computing EstimatedRTT, the TCP RTO interval is set to that value plus a safety margin in order to avoid any unnecessary retransmissions and large data transfer delay.

$$RTO = Estimated\ RTT + 4\, DevRTT \qquad \text{Equation 2.3}$$

2.6 TCP CONGESTION CONTROL IN THE INTERNET

With the fast development of the network, more and more networks access the Internet. The Internet has been expanded in terms of its scale, coverage and users quantities. More and more users use the Internet as their data transmission platform to implement various applications. Apart form traditional applications of World Wide Web (WWW), e-mail and file-transfer protocol (FTP), network users try to expand some new applications, such as tele-education, video telephone, video conference and video-on-demand (VoD), on the Internet. A best-effort network like the Internet does not have the notion of admission control or resource reservation to control the imposed network load, i.e., the total number of packets that can reside within the network. A best-effort network under high network load is called *congested*.

If the network becomes congested, no one can use the network resources at all and also the fact that when the network is congested, any additional transmitted packets would be lost because of lack of network resources such as the buffer spaces at the routers. So, network end-points sharing a best-effort network need to respond to congestion by implementing *congestion control* in order to avoid further packet drop. Otherwise, it may cause the following negative effects:

- Increase the delay and jitter of packet transmission

- Packet retransmission caused by high delay

- Decrease the network throughput and lower the utilization of network resources

- Intensified congestion can occupy too many network resources and the irrational assignment of resources even can lead to *congestion collapse*: the network load stays extremely high but throughput is reduced to close to zero [RFC 896].

The main objective of TCP's congestion control is to limit the sending rate to avoid overwhelming the network when it faces congestion on the path to the destination.

Let us first examine how a TCP sender limits the rate at which it sends traffic into its connection. Each side of a TCP connection consists of a receive buffer, a send buffer and several variables, such as LastByteRead, RcvWindow and so on. The TCP congestion control has each side of a connection keep track of an additional variable, the congestion window (CWND). The CWND size imposes a constraint on the rate a TCP sender can send traffic into the network. Specifically, the amount of unacknowledged data at a sender may not exceed the minimum of CWND and RcvWND (receiver window).

$$\text{LastByteSent} - \text{LastByteAcked} \leq \text{MIN}(CWND, RcvWND) \qquad \text{Equation 2.4}$$

TCP controls the rate of transmission of the packets as well as the congestion occurrence in the network. Therefore, the throughput of the TCP becomes a function of the size of the congestion window W and the RTT. If the throughput is measured in bytes per second, then with MSS bytes in each segment, the TCP throughput will be expressed.

$$\text{TCP Throughput} = (W * MSS)/RTT \qquad \text{Equation 2.5}$$

Let us next consider how a TCP sender perceives that there is congestion on the path between itself and the destination. A *loss event* at a TCP sender is defined as the occurrence of either a timeout or the receipt of three duplicate ACKs from the receiver. When there is an excessive congestion, one (or more) router buffers along the path overflows, causing a datagram to be dropped. The dropped datagram, in turn, results in a loss event at the sender, either by a timeout or the receipt of three duplicate ACKs, which is taken by the sender to be an indication

of congestion on the sender-to-receiver path. Notice that TCP congestion control algorithm does not require any support of routers for their functioning.

TCP congestion algorithm has three major components: additive increase and multiplicative decrease, *slow-start* and reaction to timeout events.

2.6.1 ADDITIVE-INCREASE, MULTIPLICATIVE-DECREASE

A TCP sender additively increases its rate when it perceives that the end-to-end path is congestion free, and multiplicatively decreases its rate when it detects (via a loss event) that the path is congested. For this reason, TCP congestion control is often referred to as an additive-increase, multiplicative-decrease (AIMD) [CHIU D. M & JAIN R, 1989]. The rationale for an increase in rate when it perceives no congestion is that if there is no detected congestion, then there is likely to be available bandwidth that could be additionally uses by TCP connection. In such circumstances, TCP increases its CWND slowly, cautiously probing for additional available bandwidth in the end-to-end path: it does increment its CWND a little each time it receives an ACK, with the goal of increasing CWND by 1 MSS every RTT [RFC 2581].

$$CWND = CWND + MSS\ (MSS/CWND) \qquad \text{Equation 2.6}$$

The linear increase phase of TCP's congestion control protocol is known as congestion avoidance (CA). The value of CWND repeatedly goes through cycles during which it increases linearly and then suddenly drops to half its current value (multiplicative-decrease) when a loss event occurs, giving rise to a saw-toothed pattern in long-lived TCP connections, as shown in Figure 2.4.

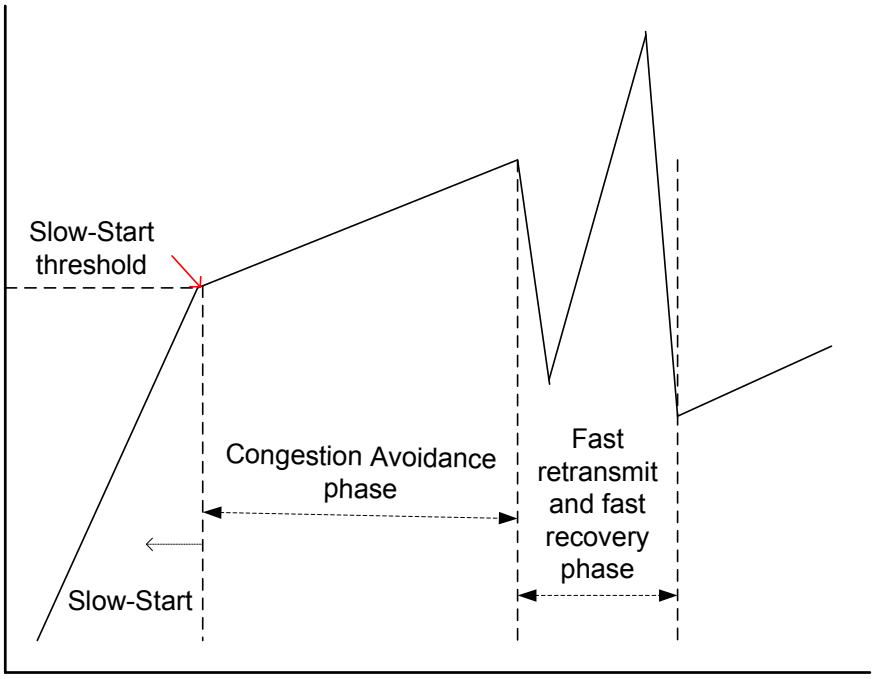

Figure 2-4 Additive-increase, multiplicative-decrease congestion control

2.6.2 SLOW START

When a new TCP connection is established with a host on another network, the CWND is initialized to 1 MSS [RFC 3390] and slow-start threshold (SSTHRESH), which determines the CWND size at which the slow-start will end and congestion avoidance will start, is set to a large value, such as equal to 65 Kbytes as in [RFC 2001, 1997]. Because the available bandwidth to the connection may be much larger than MSS/RTT, a TCP sender, during its initial phase, increases its rate exponentially by doubling its CWND value every RTT; the sender generates the exponential growth by increasing the CWND value by 1 MSS every time a transmitted segment is acknowledged.

$$CWND = CWND + MSS \qquad \text{Equation 2.7}$$

The exponential growth continues until there is a loss event, at which time CWND is cut in half, or the *SSTHRESH* is reached.

2.6.3 REACTION TO TIMEOUT EVENTS

If an ACK for a given segment is not received in a certain amount of time, known as RTO value, a timeout event occurs and the segment is resent [RFC 793]. After a timeout event, a TCP

sender enters a slow-start phase; it sets the CWND to 1 MSS and then grows the congestion window exponentially until CWND reaches *SSTHRESH*. When CWND reaches *SSTHRESH*, TCP enters the CA phase, during which CWND ramps up linearly as described in Section 2.6.1. Assuming initial value of CWND equals *1MSS*, initial value of *SSTHRESH* is large, i.e 64 Kbyes, and TCP sender begins in slow stat state, a visual description of slow start and congestion avoidance [RFC 2581] followed by a timeout is shown in Figure 2.5.

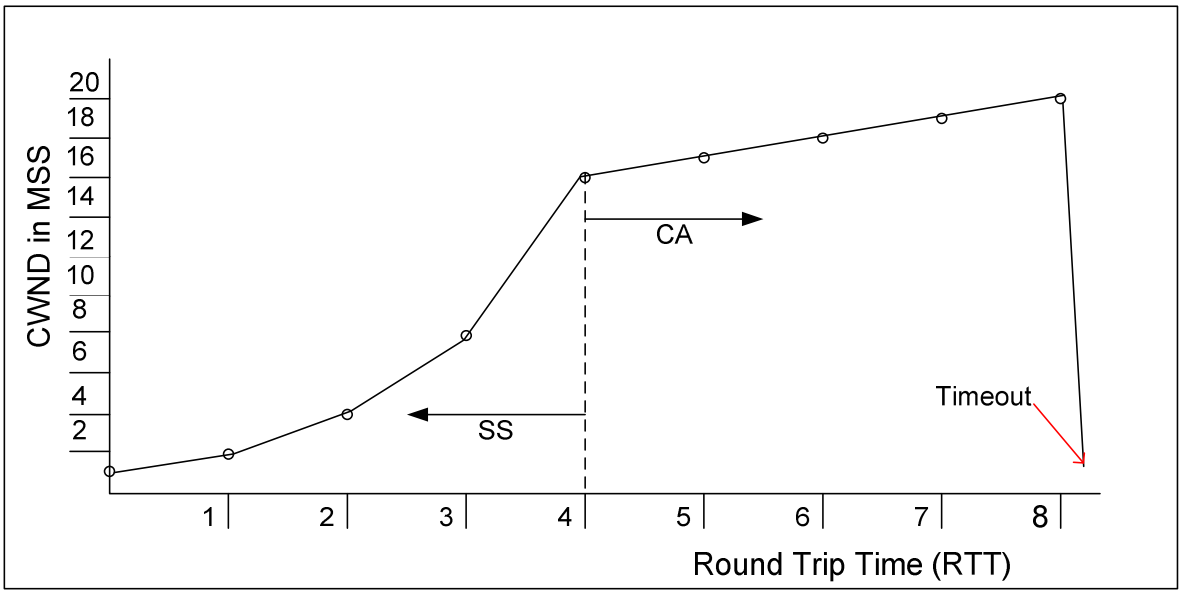

Figure 2-5 TCP Congestion Control

The TCP is based on the notion of the sliding window in order to guarantee reliable and in order packet delivery. All major types of TCP employs congestion control algorithms. However, the implementation of the fast recovery and retransmit mechanism is quite different.

2.7 TCP TAHOE

The first and most basic step taken by Van Jacobson in [Jacobson, 1988] when he established the fundamental algorithms for congestion avoidance and control. It has slow start, congestion avoidance and Fast Retransmit (FR) algorithms and was first implemented in 1988 as 4.3 BSD Tahoe TCP and was later improved in [Jacobson, 1990]. One of the major concepts behind his ideas is that a connection operating at a stable state using a full window of in transit packets should obey the *packet conservation rule*: a new packet is not put into the network until an old packet leaves the network. This implies that a packet can enter the network on the receipt of an

acknowledgement which indicates that another packet has indeed left the network. Therefore, a connection at equilibrium will send new packets with the rate it receives acknowledgements.

In FR algorithm, TCP Tahoe retransmits the lost packet upon receiving three duplicate ACKs without waiting for retransmit timer to expire. It then sets the *ssthresh* to *CWND/2* and the CWND to 1 MSS and enters into the SS phase. Later, it was found that this algorithm works well for a single packet drop but fails in case of multiple packet drops within a window of data [Sally Floyd, 1994 October]. Each retransmission of packet will force TCP Tahoe to enter SS phase thus resulting in serious loss of performance. This weakness of TCP Tahoe is partially improved in TCP Reno [Jacobson, 1990] by introducing *Fast Recovery* algorithm.

2.8 TCP RENO

TCP Reno introduces a major improvement to TCP Tahoe by modifying the action after the detection of a loss through duplicate ACKs. The idea is that the only way for a loss to be detected via a TCP timeout and not via the receipt of duplicate ACKs is when the flow of packets and acknowledgements has completely stopped. This would be an indication of heavy congestion. However, when the flow of acknowledgements is not stopped the sender should not fall back to *slow-start*. This is the case when a loss is signaled by the receipt of three duplicate ACKs rather than a timeout. Since the congestion experienced is not heavy and the flow still exists, the sender can continue with transmission but should reduce the usage of network resources.

This is implemented in the Fast Recovery algorithm that follows a *Fast Retransmit*.

TCP Reno after retransmits the missing packet by its fast retransmission algorithm will enter the *Fast Recovery* algorithm until the receipt of non duplicate ACK. Its *Fast Recovery* algorithm sets the CWND to half the *flight-size* after *Fast Retransmission* instead of reducing it to one MSS and entering into *slow-start* phase as done in TCP Tahoe. TCP Reno did significantly improve the behavior of TCP Tahoe when a single packet is dropped within a window of date. However, it suffers from multiple packet recovery within a window [Fall & Floyd, 1996].

2.8.1 TCP RENO FAST RETRANSMIT AND FAST RECOVERY

On receiving the third duplicate ACK, TCP Reno sender retransmits the missing packet by performing its *Fast Retransmit* algorithm without waiting for the expiry of retransmission timeout interval and enters into its Fast Recovery phase. It then waits for an ACK that acknowledges the

entire transmit window of data before returning to *congestion avoidance*. If it does not receive such an ACK, TCP Reno experiences a timeout and enters the *slow-start* state. The implementation of TCP Reno *Fast Retransmit* and *Fast Recovery* mechanism [RFC 2581] is as follows:

1. *ssthresh* is set to no more than *max*(*flightsize*/2, 2*MSS*) when receiving the third duplicate ACK.

2. Retransmit the lost segment and set *CWND* = *ssthresh* +3*MSS* to inflate the congestion window artificially by three that are buffered at receiver side.

3. Increment CWND by MSS for each additional duplicate ACK received to inflate the congestion window in order to reflect the additional segment that has left the network.

4. Transmit a new segment if allowed by the new value of CWND and the receiver's advertised window.

5. When the next ACK arrives that acknowledges new data, set *CWND* = *ssthresh*, which is set in step 1.

It should be noted that after the step 5, TCP Reno will enter the *Fast Retransmit* phase again instead of *Congestion Avoidance* phase if there is multiple packet losses within that window of data. This may also require a timeout to recover the lost packets, and it has been shown in [Sally Floyd, 1994 October] that TCP Reno *Fast Recovery* is generally not efficient to this effect.

2.9 TCP NEW RENO

TCP New Reno [S. FLOYD & HENDERSON, 1999] introduces a small enhancement to TCP Reno. In simple, TCP Reno sender would leave *Fast Recovery* on the receipt of first ACK that acknowledges new data. This works fine if there is only one lost packet. However, when more losses exist this will fail to recover all of them. [Hoe, 1996] suggested that during the *Fast Recovery* the TCP sender should respond to partial ACK by inferring that the indicated packet has been lost and retransmitting that packet. TCP New Reno, contrary to TCP Reno, does not exit its *Fast Recovery* phase on receiving partial ACKs. Instead, it retransmits that indicated lost packet on the arrival of each partial ACK, thereby recovering from multiple packet loss in a single window of data and exits its *Fast Recovery* phase either on receiving the ACK that acknowledges entire data within that window or on occurrence of retransmission timeout.

2.9.1 TCP NEW RENO FAST RECOVERY ALGORITHM

The TCP New Reno Fast Recovery algorithm described in [S. FLOYD & HENDERSON, 1999] is as follows.

1. Initialize a new variable *send_recover* to initial send sequence number. When the third duplicate ACK is received and the sender is not already in the Fast Recovery phase, check that the duplicate ACKs cover more than variable send_recover. If they do, then set ssthresh to max(*flightsize*/2, 2*MSS*) and record the highest sequence number transmitted in the variable *send_recover*.

2. Steps 2, 3 and 4 of TCP Reno given in Subsection 2.8.1 are processed.

3. When the next ACK arrives that acknowledges all data packets up to and including *send_recover*, then set CWND either to min(ssthresh, flightsize + MSS) or ssthresh as in step 1 and exit the Fast Recovery phase. Otherwise, this indicates a partial ACK. Do not exit Fast Recovery and performs the followings

    a. Retransmit the first unacknowledged packet, deflate CWND by the amount of new data being acknowledged and add one MSS and reset the retransmit timer.

    b. It then tries to send new segment depending on the new CWND size and the receiver window size.

4. If a retransmit timeout happens then it records the highest sequence number transmitted in the variable *send_recover*, exits *Fast Recovery* phase.

2.10 TCP SELECTIVE ACKNOWLEDGEMENT

A proposed modification to TCP, selective acknowledgement (SACK) [RFC 2018] allows a TCP receiver to acknowledge out-of-order segments selectively rather than just cumulatively acknowledging the last correctly received in-order segment. SACK option is used by TCP receiver to inform the TCP sender that a non contiguous segment of data has been received and it is queued. To use SACK, both the TCP sender and receiver must support the feature and must enable it by negotiating the SACK-Permitted option during the connection establishment.

Adding the SACK option to the TCP flavors such as TCP Tahoe or Reno, does not change their basic underlying congestion control algorithms. The information about missing sequence

numbers is transmitted to TCP sender using three SACK blocks with each ACK, using the rules outlined in [RFC 2018]. A simulation based comparison of TCP Tahoe, TCP Reno and TCP SACK [Fall & Floyd, 1996] showed that TCP SACK recovers from multiple packet losses quickly and smoothly without the expiry of retransmission timeout interval. Further, real time Internet experiments in [Bruyeron, Hemon, & Zhang, 1988] shows that depending on the error pattern, TCP SACK has 10% to 45 % higher throughput than TCP Reno.

2.11 TCP FORWARD ACKNOWLEDGEMENT

The Forward Acknowledgment (FACK) algorithm proposed in [RFC 2018] aims at better recovery from multiple losses. In FACK, TCP maintains two additional variables: *snd_fack* that represent the forward-most segment that has been acknowledged by the receiver through the SACK option and *retrans_data* that reflects the amount of outstanding retransmitted data in the network. Using these variables, the sender can estimate the actual quantity of outstanding data in the network as (forward-most data sent - snd_fack + retrans_data) and can inject new data if allowed by receiver's window. TCP FACK regulates the amount of outstanding data in the network to be within one segment of CWND, which remains constant during the *Fast Recovery* phase.

2.12 TCP VEGAS

TCP Vegas [Brakmo, O'Malley, & Peterson, 1994] was presented before New Reno, SACK and FACK were developed. Vegas is fundamentally different from other TCP variants in that it does not wait for loss to trigger congestion window reductions.; it employs an alternative strategy in that it tries to predict when congestion is about to happen and adapts its window to compensate. This is a proactive approach as it attempts to reduce its sending rate before packets start being dropped by the network.

Vegas keeps track of the time each segment is sent. When an ACK arrives, it estimates RTT as the difference between the current time and the recorded timestamp for the relevant segment. For each connection, Vegas defines BaseRTT to be the minimum RTT seen so far. The actual throughput of TCP Vegas is obtained by dividing the number of bytes sent in a round trip time (RTT) and its expected throughput is obtained by dividing current CWND size by the minimum RTT. TCP Vegas does not look at changes in the slope of throughput but alternatively it compares the actual throughput with expected throughput to determine change in its congestion window size.

TCP Vegas uses the fact that as the CWND size increase the throughput of the connection should also increase to measure and control the amount of extra data the connection should have in transit; sending too much data will obviously cause congestion, thereby increasing the RTT. However, it is also important for Vegas to maintain enough throughput to actually allow it to adjust the throughput to the available bandwidth.

2.13 TCP PERFORMANCE EVALUATION

TCP performance is mainly determined by its throughput and fairness. However, other factors such as link utilization and packet loss rate are also important in effective evaluation of TCP.

The throughput performance is defined as the total number of original packets received by the receiver in a given period of time. TCP throughput is influenced by many parameters. The granularity of the TCP timers can vary from 10ms to 500ms, depending on the TCP implementation of the host system [Stevens & Wright, 1994]. These implementation specific details can result in a considerable throughput discrepancy of different variants under the same network conditions.

Fairness is an important performance criterion in all resource allocation schemes. Its most intuitive and obvious definition is that all sessions of data flow should be entitled to use the network resources including the bandwidth equally without any bias. Thus a fair TCP connection does not deprive other connections their fair share of bandwidth [BERTSEKAS & GALLAGER, 1996].

2.14 TCP FOR WIRELESS CHANNEL

The properties of wireless channels are very different from those of wired channels. Wireless channels are characterized by high bit error rate with random losses caused by shadowing and fading. Furthermore, the channel may cause burst errors when the channel is in a deep fade for a significant amount of time or when the channel is in short fade where the length of burst error could vary. The low efficiency of TCP in a wireless channel is due to the fact that it misinterprets packet losses due to high error rate and congestions. However, because of the common usage of the TCP in Internet applications that require reliable data transfer, it is important to keep the TCP/IP protocol stack and also the network element structure as unchanged as possible even when mobility features required in wireless Internet are added to the network [Jamalipour, 2003].

It emphasize that the TCP should be modified in order to meet the TCP performance expectation in wireless channel.

2.15 CONCLUSIONS

In this Chapter, we have outlined the development of TCP and described its main features with an emphasis on congestion control and packet loss recovery mechanism. Major types of TCP variants and their specific functionalities are explained; we have seen how packet losses are detected, retransmitted and congestion window is calculated depending on the selected TCP variants. Specially, the *Fast Retransmit* and *Fast Recovery* algorithms of TCP Reno and TCP New Reno are described in details.

Furthermore, it is outlined that in the wireless channels the main cause for packet loss is the high bit error rate and not the network congestion.

Chapter III.

Review of TCP Enhancements for Wireless Networks

Wireless networks are becoming increasingly popular in the world of telecommunication. As a consequence, a significant effort has been devoted to the provisioning of reliable data delivery for a wide variety of applications over different wireless infrastructures. One of the major challenges in modern communication system is to provide wireless access to the Internet. TCP supports the most popular suit of applications on the Internet today and it has been enhanced in recent years to improve robustness and performance over network of varying capacities and quality. However, it largely retains the behavior outlined in [RFC 793] including properties that make it a less suitable transport protocol for wireless medium.

In this Chapter, we give an overview of some optimizations that have been proposed in the literature and describe how they differ in terms of their retransmission and recovery mechanisms. The proposed optimizations can be categorized into four groups; split-connection, link-layer, explicit notification and end-to-end protocols. Section 3.1 describes how optimization at the transport layer can be achieved by splitting the connection at the base station. Link-layer optimizations for improved TCP performance are presented in Section 3.2. Explicit notifications can be used between an intermediate node and the end hosts in order to distinguish the congestion related losses from the wireless error and are outlined Section 3.3. The End-to-End approaches do not require any intermediate node's support and are described in Section 3.4. Motivated by these criteria, we propose new techniques to enhance the TCP performance over a wide range of network and traffic configurations in Section 3.5.

3.1 SPLIT-CONNECTION PROTOCOLS

The idea of split-connection approaches is to divide each TCP connection into two separate connections at an intermediate node; one is between the fix host (FH) and the base station (BS)

and the other between the BS and mobile host (MH) as shown in Figure 3.1. Such a subdivision of the traditional end-to-end connection offers several advantages [Fieger & Zitterbart, 1997].

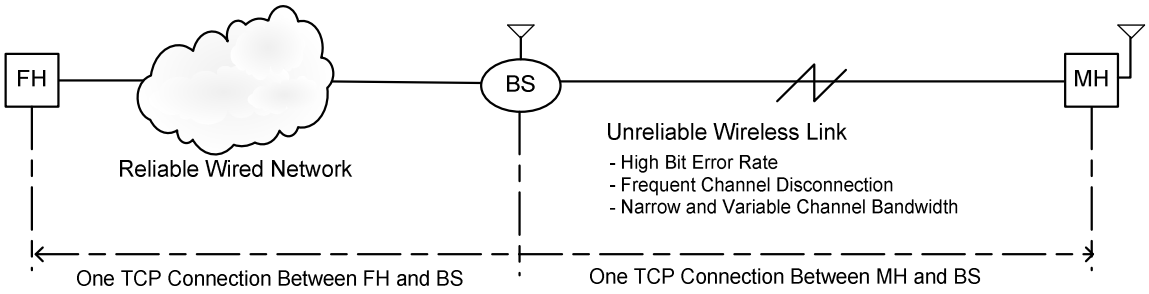

Figure 3-1 Splitting the TCP Connection into two separate connections

The transmission characteristics of the wireless link such as high bit error probability and disruptions due to radio shadow or handoffs influence only the transport connection over the wireless hop. This way any effects due to wireless link can be hidden from nodes within the wire-line network and makes it possible for the wire-line and wireless medium to be employed with different transport protocols. However, since TCP connections are terminated at the base station, data buffers and TCP state information should be maintained at the base station in any split-connection scheme. When a TCP connection is created, the socket send buffer and receive buffer are allocated. The buffer size can be specified by the process that creates the socket or a default value can be used. In either case, the TCP send and receive buffer sizes are fixed for the duration of the connection.

3.1.1 INDIRECT TCP

Indirect TCP (I-TCP) [Bakre & Badrinath, 1995] is a split-connection solution that utilizes the resources of mobility support routers (MSRs) to provide transport layer communication between mobile hosts and fixed hosts.. It uses the standard TCP for its connection over the wireless hop and, like other split-connection protocols, attempts to separate loss recovery over the wireless link from the wired link. It ensures that packet errors and delay variations on the wireless link do not result in the initiation of TCP congestion control procedures or affect the TCP retransmission timer on the wire-line connection, and eliminates the end-to-end retransmission of packets that suffer error across the wireless link. An experiments in [Balakrishnan, Padmanabhan, Seshan, & Katz, 1997] indicates that the choice of TCP over the wireless link

results in inefficient utilization of the wireless link capacity and adds overhead to the base station. This shortcoming is addressed in two other split-connection protocols, Mobile-End Transport Protocol [Kuang-Yeh & Tripathi, 1998] and Mobile-TCP [Haas & Agrawal, 1997].

3.1.2 SPLIT-CONNECTION WITH SELECTIVE REPEAT PROTOCOL

The authors in [Yavatkar & Bhagawat, 1994] proposes a split-connection protocol that introduces a new session layer protocol on top of TCP at both base stations and mobile hosts to compensate for effects of wireless link characteristics and host migration. The session layer protocol is designed to exploit the available knowledge about both wireless link characteristics and host migration and to compensate for highly unpredictable and unreliable between a mobile host and its base station. The intermediate agent at the base station participates in the session layer protocol and forwards incoming traffic over a TCP connection to the remote mobile host. It proposes two transport protocols over the wireless link; one is the standard TCP and the other one a selective repeat protocol (SRP) on top of user datagram protocol (UDP).

SRP implements its own flow and error control mechanisms designed and optimized specifically to handle the wireless link effects. It also uses a selective repeat algorithm in which a receiver returns a selective ACK (SACK) when an out of order segment is received. The SACK specifies the missing segments using a bitmap that includes the sequence numbers of the latest segment and the last in order segment received. Using this alternative, unlike the TCP, SRP enables the sender to recover from multiple losses within a window of data, thereby increasing the throughput performance over the wireless link. However, their study in the impact of handoffs on performance concludes that the use of SRP instead of TCP as the transport protocol over the wireless hop does not obtain any significant advantage.

3.1.3 MOBILE-END TRANSPORT PROTOCOL

Mobile-End Transport Protocol (METP) [Kuang-Yeh & Tripathi, 1998] proposes to eliminate the TCP and IP layers from mobile hosts. A mobile host will replace TCP/IP headers of the packets transmitted over a wireless link with a header containing essentially only de-multiplex keys and the source and destination IP addresses. METP considers that the hop between a mobile host and its base station is either the first or the last one along a data path and the mobile host does not perform datagram forwarding. Hence, only a part of the IP functionality needs to be shifted to the base station. All TCP connections are handled at the base station by METP on behalf of the mobile host; it negotiates with another host in the Internet to establish or close a

TCP connection. When a TCP segment destined for the mobile host arrives at the base station, METP puts it in the receiving buffer and sends an acknowledgement back to the source thereby any congestion control or congestion avoidance mechanism of TCP reflects only the state of the wire-line part of the connection.

Since the TCP connection is terminated at the base station, METP has to provide reliable in-order delivery over the wireless link. It adds tremendous overhead to the base station and makes the mobile host becomes more dependant on the base station. For example, when a packet has arrived at and been acknowledged by the base station, the sender will confirm that that packet has been successfully transmitted. However, if the base station fails thereafter, that packet may never arrive at the mobile host. In such a case, the mobile host should take appropriate recovery measures as the failure were happened to itself. This will further increase the buffer size, thereby adding more overhead to the base station.

3.1.4 MOBILE-TCP

Mobile-TCP [Haas & Agrawal, 1997], employs header compression to reduce the amount of wirelessly transmitted data. It considers that there is no need to communicate with the full TCP-layer source and destination addresses as all the packets form the MH pass through the BS. During the connection establishment process, a connection ID (CID) is assigned to each direction and it is used in any future exchange of data over the wireless segment. The CID information, which includes the source and destination IP addresses and the corresponding port numbers, are cached at both the MH and BS. When MH sends a packet to the network, the TCP address is translated into the corresponding CID, which is then expanded back into the TCP-layer address at the BS. Similar operation is performed in the reverse direction. In Mobile-TCP, TCP segments originated in the fixed host are acknowledged by the TCP entity in the base station only when successful wireless transmissions of the segments are acknowledged by the mobile host. Mobile-TCP thus preserves the end-to-end semantics of the TCP, but the downlink TCP data flow is affected by variable delays in the wireless link.

3.1.5 SPLIT-CONNECTION MOBILE TRANSPORT PROTOCOL

Split-Connection mobile transport protocol (SCMTP) [Xie, Hammond, & Noneaker, 2003] proposes a scheme similar to the mobile end transport protocol (METP) described in Section 3.1.3. In common with METP, SCMTP employs a standard TCP protocol on the wire-line connection between the fixed host and base station, and a light weight transport protocol over

the wireless link. However, it employs an automatic repeat request (ARQ) protocol to handle the wireless error and uses a different channel access protocol; time division multiplexing (TDM) is used to allocate the forward-link (BS to MH) capacity to each mobile host, and a time-division multiple access (TDMA) is employed by each mobile host on the reverse-link. If there are multiple traffic between the base station and a given mobile host, scheduling algorithms in the base station and the mobile host determine how the flows share the forward-link and reverse-link capacities respectively. Their experiment concludes that the use of Go-back-N ARQ can exploit the wireless link capacity more efficiently than that of stop-and wait ARQ protocol.

3.1.6 CONCLUSION

The schemes presented above have tried to improve the performance of TCP over wireless network by shielding the sender from wireless effects. None of these schemes actually lets the TCP sender know clearly whether the packet is lost because of network congestion or wireless error. This makes the TCP sender retransmit the packet as usual, subsequently being unable to keep the throughput high in the error-prone wireless environment. They also violate the end-to-end semantics of TCP. They also maintain a significant amount of TCP state at the base station per TCP connection that makes the handoff procedures slow and complicated.

3.2 LINK-LAYER PROTOCOLSS

Link-layer protocols are another alternative for improving the poor performance of TCP over wireless links. The concept behind this is to make the wireless link layer look similar to the wired case for TCP by recovering the wireless error locally. There have been several proposals for reliable link-layer protocols. The main techniques employed by these protocols are forward error correction (FER) and automatic repeat request (ARQ). This method is illustrated in Figure 3.2. ARQ is frequently used in data communications protocols. When a frame is detected to contain errors after decoding, it is discarded and an ACK is sent back to the sender requesting a retransmission of the frame. This is called a selective retransmission and the most efficient way of retransmission. Cellular networks such as GSM/GPRS and UMTS have recently incorporated the concept of ARQ in order to improve performance of data transfers over wireless link [Ladas, Amiee, Mahdavi, & Manson, 2002]. We describe some of these link-layer proposals and conclude their ability to improve the TCP performance over wireless link.

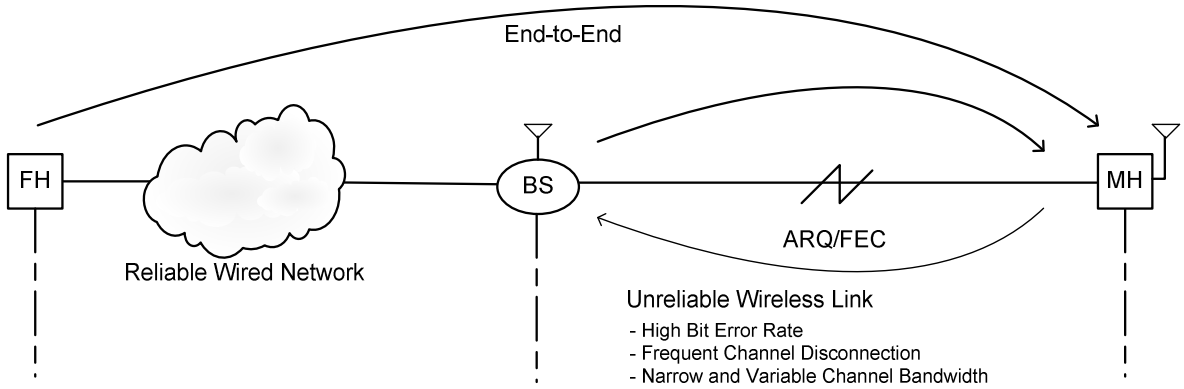

Figure 3-2 A link-layer approach to improve the TCP performance

3.2.1 SNOOP PROTOCOL

The Snoop protocol [Balakrishnan, Padmanabhan, Seshan, & Katz, 1997] is a TCP-aware link layer protocol. It uses link layer retransmission to improve the reliability of the wireless link, and actively tries to avoid unnecessary TCP retransmissions. In this method, the base station is equipped with a module called *snoop agent*, the functionality of which is to monitor the TCP packets transmitted from a fixed host to a mobile host and vice versa. The agent caches all these packets locally and uses this information to detect wireless packet losses and timeouts. In the case of detecting a wireless packet loss, it retransmits the packet promptly and suppresses the duplicate ACK reaching the TCP sender. This way, it prevents the sender from invoking unnecessary fast retransmissions and congestion control algorithms.

3.2.2 ADAPTIVE-TCP PROTOCOL

The Adaptive TCP (A-TCP) is a TCP aware link layer protocol and maintains end-to-end semantics of TCP. The basic concept of the protocol is to make a wireless link look like a wired link by employing an A-TCP agent in the base station. This is referred to as a virtual host model. The A-TCP agent implements three basic functions, such as local retransmissions, sender freezing and A-TCP flow control to hide the wireless environment form the fixed host. The local retransmission diminishes the effect of high bit errors. On receiving duplicate ACKs, the A-TCP agent, similar to the Snoop agent, filters the duplicate ACKs and locally retransmits the lost packet. It also keeps a retransmission timer of the standard TCP sender. When it expires for a particular segment, the A-TCP agent immediately retransmits that segment. Long-term channel disconnections are handled by the use of sender freezing [Goff, Moronski, Phatak, & Gupta, 2000]. The A-TCP flow control is the main factor for improving the TCP performance in a

wireless environment. In the A-TCP flow control, the A-TCP agent marks the window field of acknowledgement segment with a retransmission buffer size, thereby avoiding wireless link overflow. Thus, the TCP congestion control at the sender will not be triggered by wireless link overflow.

3.2.3 ASYMMETRIC RELIABLE MOBILE ACCESS IN LINK-LAYER (AIRMAIL) PROTOCOL

The AIRMAIL protocol [Ender Ayanoglu, Sanjoy Paul, Thomas F. LaPorta, Krishan K. Sabnani, & Gitlin, February 1995] provides a reliable link layer in conjunction with forward error correction. In this approach, in order to conserve bandwidth power, the base station sends an entire window of data before an ACK is returned by the mobile receiver. A consequence of this approach is that there is no opportunity to correct errors until the end of an entire window, which can cause TCP to time out if the error rate is large or cause a large variation in delay depending on the position of the loss within the window. Another approach [Chaskar, Lakshman, & Madhow, 1996], which demonstrates the validity of link-layer solutions, shows analytically how to achieve throughput by insuring the buffer at the interface to the wireless connection is sufficient. In AIRMAIL, a simple Stop-and-Wait protocol is used over the wireless link to quickly retransmit packets before TCP discovers the loss.

3.2.4 RADIO LINK PROTOCOL

The Radio Link Protocol (RLP) is a link layer protocol that, like TCP, provides a reliable byte stream service and typically used over cellular networks. RLP fragments the TCP segment into frames and uses robust error correcting codes and fast retransmission schemes to shield the wireless channel related losses from the TCP sender, thus preventing TCP throughput degradation. The fragmentation is done to increase the granularity of the transmission. In case of any error, an RLP frame which is of a smaller size is affected rather than the entire TCP segment. The RLP uses an ARQ error recovery mechanism to retrieve a lost RLP frame, which can be an acknowledgement based (ACK-based) or negative acknowledgement based (NACK-based). Since the reverse link is very expensive on most cellular networks, most RLPs implement NACK-based scheme in which the recovery process is initiated by the receiver by requesting a retransmission of only the missing or erroneous frame.

RLP's error recovery persistency can be configured via a parameter that defines the maximum number of retransmission of a single frame. When the RLP sender detects that a frame could not

be transmitted successfully after all the requires-retransmission attempts, it not only discards that frame, but also resets the link, i.e, re-initializes the sequence numbers. In this case, the corresponding TCP segment will be discovered at the TCP layer.

3.2.5 TULIP PROTOCOL

The transport unaware link improvement protocol (TULIP) [Parsa & Garcia-Luna-Aceves, 1999] is, similar to the Snoop protocol, to improve the TCP performance over lossy wireless links without the need to modify the transport layer protocol. However, it does not require a proxy at the base station and keeps no TCP states. TULIP provides reliable service for TCP data traffic and an unreliable service for UDP and TCP ACKs. TULIP does not provide reliable service to TCP ACKs because subsequent ACKs supersede the information in the lost ACK. The receiver simply buffers packets and passes them up to the next layer in order, thereby preventing TCP from generating duplicate ACKs in the event that a packet is missing from the expected sequential packet stream. This ability of TULIP to maintain local recovery of all lost packets at the wireless link in order to avoid the unnecessary and delayed retransmission of packet over the entire path. TULIP maintains timers that rely on a maximum propagation delay over the link, rather than performing a round-trip time estimate of the channel delay.

3.2.6 CONCLUSION

The proposal described above use link layer retransmission to minimize packet loss due to the wireless part of the connection. Link layer protocols focus on the problems that arise from lossy wireless links. The main advantage of employing a link layer protocol for wireless loss recovery is that it fit naturally in the layered structure of the network protocols. The link layer protocol operates independently of higher layer protocols, and does not maintain any per-connection state. All link layer proposals preserve End-to-End semantics of TCP, but TCP data acknowledgements are required to pass through the same base station. The main concern about the link layer protocols is the possibility of having an inverse effect on certain transport protocols such as TCP. Independent timer reaction at link and transport layers that may result in unnecessary retransmissions, fast retransmission interaction, and large round-trip variations are considered as major problem with link-layer approaches [Balakrishnan, Padmanabhan, Seshan, & Katz, 1997].

3.3 EXPLICIT NOTIFICATION

A various explicit notification schemes have been proposed to enable the TCP sender to distinguish between different types of packet losses. Examples of this approach include Explicit Congestion Notification (ECN) [Ramani & Karandikar, 2000], Explicit Loss Notification (ELN) [Balakrishnan, Padmanabhan, Seshan, & Katz, 1997]. The idea behind this approach is to enable the TCP sender to distinguish packet losses due to congestion from wireless error.

3.3.1 EXPLICIT CONGESTION NOTIFICATION

In this method, a TCP receiver informs the TCP sender of the network congestion explicitly through one of the reserved bits in the TCP header, called the ECN-Echo (ECE) flag, when it receives an IP packet with the congestion experienced (CE) bit in the IP header set. ECN is an extension proposed to random early detection (RED) [S. Floyd & Jacobson, 1993], which monitors the average queue size and marks packets instead of dropping them based on statistical probabilities. Since ECN marks packets before congestion actually occurs, this is useful for protocols like TCP that are sensitive to even a single packet loss. The packets provided with ECN support is referred as ECN capable packets.

ECN requires support from both the routers as well as the end hosts. It requires the routers to be able to identify packets that are ECN capable and to mark only such packets from ECN capable hosts. Two bits in the IP header field are utilized to achieve this; the ECN Capable Transport bit (ECT) is set by the sender during the connection establishment process if both the end systems are ECN capable. The CE bit of the packets encountering congestion is marked by the router on their way to the receiver with a probability proportional to the average queue size used in RED. It also proposes to add two new flags, namely the ECE flag and congestion window reduction (CWR) flag, in the reserved field of TCP header as shown in Figure 3.3.

<table>
<tr><td colspan="8">Source port number (16-bits)</td><td colspan="2">Destination port number (16-bits)</td></tr>
<tr><td colspan="10">Sequence number</td></tr>
<tr><td colspan="10">Acknowledgement number</td></tr>
<tr><td>4-bit Header length</td><td>Reserved (4 bits)</td><td>CWR</td><td>ECE</td><td>URG</td><td>ACK</td><td>PSH</td><td>RST</td><td>SYN</td><td>FIN</td><td>Receiver window size (16-bits)</td></tr>
<tr><td colspan="8">Checksum (16-bits)</td><td colspan="2">Urgent Data (16-bits)</td></tr>
<tr><td colspan="10">Options</td></tr>
</table>

Figure 3-3 TCP Header with ECN and CWR Flags

The ECE flag, which is set by the TCP receiver, indicates congestion in the network. On receiving an ACK with the ECE flag set, the sender sets the CWR flag to inform the receiver that it has reacted to its congestion notification.

3.3.2 EXPLICIT LOSS NOTIFICATION

On the basis of the performance improvement achieved in TCP Snoop and ECN protocols, a new protocol, namely, explicit loss notification with acknowledgement (ELN-ACK) [Wenqing & Jamalipour, 2001b] is proposed that could remedy the limitations of the Snoop protocol [Balakrishnan & Katz] [Wenqing & Jamalipour, 2001a, , 2001b] . In the ELN-ACK scheme, a new form of acknowledgement packet called ACK_{ELN} is used to communicate the cause of packet losses to the TCP sender and no packets are cached at the base station. An ELN-ACK agent, similar to the Snoop agent, is introduced at the base station to perform two main features.

- One is to judge and store the packet loss information transmitted form the fixed host. If the base station receives an out of order packet from the fixed host, it will store the corresponding packet information in the ELN-ACK agent.

- On receiving an ACK_{ELN}, the base station will judge the lost packet based on the stored information. If it finds the packet has been lost before arriving at the base station, it will set the ELN bit to indicate the packet was lost due to congestion. Otherwise, it will reset the ELN bit to indicate the packet loss was due to wireless error.

When the sender receives an ACK with the ELN bit set, it retransmits the next segment, but does not trigger any congestion control actions. The sender also makes sure that each dropped segment is retransmitted at most once during the course of a single round trip since the agent would set the ELB flag for each duplicate ACK following a loss.

3.3.3 ICMP MESSAGING

The authors in [Goel & Sanghi, 1998] proposes an ICMP based scheme that makes the TCP sender aware of wireless errors. The base station generates two ICMP messages. One is ICMP-DEEFER message when its fist attempt in transmitting the packet over the wireless link fails. This ensures that TCP sender will receive either an ACK or an ICMP message during one round trip. A lack of both will signal a congestion loss. Thus, TCP can distinguish a packet loss due to

congestion from wireless loss. Second one is an ICMP-RETRANSMISSION message when it discards the packet after all retransmission attempts have been exhausted.

On receiving an ICMP-DEEFER message, TCP postpones the retransmission timer if the lost segment is one for which the timer is active. This will help to avoid conflict between the link layer and end-to-end retransmissions. When the TCP sender receives an ICMP-RETRANSMISSION message, which indicates that one segment was lost across the wireless link, it retransmits the segment indicated and enters the fast recovery phase. On receiving a new ACK that acknowledges the retransmitted segment, it comes out of recovery phase, resetting its CWND to the value prior to entering the fast recovery phase. When the TCP suffers a retransmit timeout or receives three supplicate ACKs for which it has not received an ICMP-RETRANSMISSION message, it follows the standard TCP procedures.

3.3.4 SYNDROME

[W.P. Chen, Y.C. Hsiao, J. C. Hou, Y. Ge, & Fitz] proposes a light-weight approach, called syndrome, to improve TCP performance in wireless environments. In syndrome, the base station counts the number of packets it has relayed to the destination host for each TCP connection and includes this in the TCP header option. The destination host will use both the syndrome counter and the sequence number to determine whether the packet is lost due to the congestion or due to wireless error. Gaps in the syndrome counter will indicate that packets were lost on the wireless link. Gaps in the sequence number but not syndrome counter will indicate that packets must have been lost due to congestion in with wire-line part of the network. Determining the cause for the packet loss, the TCP receiver notifies the sender via explicit loss notification to take appropriate action. If a packet loss is due to the transmission error on the wireless link, the sender does not reduce its congestion window.

3.3.5 MULTIPLE ACKNOWLEDGEMENTS

[Biaz & Vaidya] proposes to use two types of acknowledgements to distinguish congestion losses from wireless errors. This proposal uses one additional acknowledgement, called partial acknowledgement, which the base station transmits in response to data from the TCP sender in the fixed network. Provided that no segments are lost, the sender receives two acknowledgements for each segment; a partial acknowledgement (ACKp) from the base station and a complete ACK from mobile host. If the sender receives only the ACKp, it can deduce that

the data must have been lost over the wireless hop and no congestion control action is required. If no acknowledgements arrive, then the most likely cause for the data loss is congestion.

A similar scheme [Cobb & Agrawal, 1995] that introduces two new partial acknowledgements to improve the performance of TCP sessions that originate or terminate in noisy wireless networks for mobile hosts. If the receiver is a mobile host, the base station transmits a last hop acknowledgement in response to the fixed host. In case the mobile host is the sender, then the base station transmits a first hop acknowledgement in response to the mobile host. As in the partial acknowledgement approach described above, the sender receives two acknowledgements for each successfully transmitted segment, one from the base station and from the receiver. This acknowledgement approach allows the TCP sender to distinguish losses due to congestion and losses due to wireless errors and to take appropriate action.

3.3.6 CONCLUSION

The explicit notification proposals have a different philosophy compared to the split-connection approach. They have significantly reduced the overhead introduced at the base station and enable the TCP sender to distinguish congestion from the wireless error and to implement the fast retransmit and fast recovery mechanism that may well suit in the wireless environments. The explicit notification scheme does not solve the problem with the higher unreliability of the wireless network. However, since the sender knows about this effect, it can make a more informed decision.

3.4 END-TO-END PROTOCOLS

The standard TCP implementations rely on packet loss as an indicator of network congestion and lack the ability to distinguish congestion losses from losses invoked by noisy links. In wireless connections, overlapping radio channels, signal attenuation and additional noises have a huge impact on such losses. As a consequence, the standard TCP reacts with drastic reduction of the congestion window, thus degrading the performance of TCP. End-to-end proposals make the TCP sender handle packet losses caused by both congestion and random wireless errors and requires minimal or no processing at the base station. Another advantage of these schemes is that the end-to-end semantics of TCP is maintained. Some of the end-to-end schemes proposed in the literature are described below.

3.4.1 END-TO-END SMART

SMART (Simple Method to Aid Retransmission) protocol [Keshav & Morgan, 1997] combines aspects of both the traditional techniques Go-Back-N (GBN) and selective retransmissions (SR) [Doshi, Johri, Netravali, & Sabnani, 1993]. The key idea in SMART is to build the bit-mask of correctly received packets at the sender instead of carrying it in the ACK header. Each ACK therefore carries two pieces of information; the cumulative ACK as in the standard GBN and the sequence number of the packet that caused that ACK to be generated. The second piece of information not only allows the sender to identify which packets have been correctly received, but also enable to infer which packets have been lost and to retransmit those lost packets selectively. When the sender detects gaps in the bitmask, it immediately assumes that the missing packets have been lost without considering the possibility that they simply may have been reordered. Thus, this scheme trades off some resilience to reordering and lost acknowledgements in exchange for a reduction in the overhead to generate and transmit acknowledgements [Balakrishnan, Padmanabhan, Seshan, & Katz, 1997]. The SMART allows the sender to handle multiple losses within a window of outstanding data efficiently. However, the sender still assumes that losses are a result of congestion and invoke congestion control mechanism.

3.4.2 TCP WESTWOOD

TCP Westwood is a sender-side modification to TCP NewReno that controls the congestion window using end-to-end rate estimation and only affects the congestion avoidance algorithm and keeps the slow-start phase unchanged, as well as the linear increase in the congestion avoidance phase [Gerla et al., 2001] [Grieco & Mascolo, 2003]. The TCP sender, by monitoring the ACK reception rate, continuously estimates the packet rate of the connection and uses this estimate to determine the available bandwidth. When the sender perceives that congestion has appeared, the sender uses the estimated available bandwidth to set the congestion window and the slow-start threshold sizes. The rationale is that if a connection is achieving a given rate, then it can be safely used to obtain the corresponding CWND and threshold setting without causing congestion in the network. Adjusting the congestion window to the estimated available bandwidth makes TCP Westwood more robust to wireless losses since the CWND is not reduced to half, but it is adapted to the most recent bandwidth estimation instead, which may lead to unfairness. However, performance analysis in [Casetti, Geria, Lee, Mascolo, & Sanadidi,

2000] shows that TCP Westwood manages to obtain a fair share of the bandwidth when it coexists with other TCP Westwood connections.

3.4.3 FREEZE-TCP

Freeze-TCP [Goff, Moronski, Phatak, & Gupta, 2000] is a mechanism that places the sender in persist mode prior to a disconnection through signal strength measurements. In this method, mobile host monitors the signal strength and sends zero window advertisement (ZWA) if it detects an impending disconnection. By exploiting the properties of the receiver advertised window, a TCP connection can be frozen. If the receiver sets the receiver window to zero, then the sender leaves its CWND unchanged until the receiver advertises a new receiver window. This prevents segments from getting lost and unnecessary congestion control action to be taken by the sender. Upon reconnection detection, the receiver sends three copies of acknowledgements of last received prior to the disconnection, as in [Caceres & Iftode, 1995], in order to awake the sender and to resume the transmission at the same rate as before. A possible drawback of Freeze-TCP is that it depends on the ability of the lower layers to detect an incoming disconnection and notify the TCP sender of this in a timely manner.

3.4.4 DELAYED DUPLICATE ACKNOWLEDGEMENT

The delayed duplicate acknowledgement scheme proposed in [Nitin H. Vaidya, 2002] is an end-to-end scheme that attempts to mimic the behavior of Snoop protocol. In this method, the receiver delays the third and subsequent duplicate ACKs for a predetermined interval while the base station performs link level retransmissions. During this time, if the receiver receives the missing data, it will transmit cumulative ACKs and discards the delayed duplicate ACKs. Otherwise, it will release the delayed duplicate ACKs when the timer expires. This is an attempt to prevent the fixed host from triggering congestion control action while the base station retransmits the data over the wireless link. One disadvantage of this scheme is that if the duplicate ACKs are caused by congestion, delaying the duplicate ACKs will unnecessarily delay the error recovery process. Explicit loss notification to the receiver (ELNR) proposed in [Mehta & Vaidya] is an enhancement to the delayed duplicate acknowledgement scheme.

3.4.5 WIRELESS TRANSMISSION CONTROL PROTOCOL

The wireless transmission control protocol (WTCP) [Sinha, Nandagopal, Venkitaraman, Sivakumar, & Bharghavan, 1999] uses a rate based approach as in TCP Westwood to control the transmission rate. The ratio of the inter-packet separation at the receiver and the sender is used as

the primary metric for transmission control rather than using packet losses and retransmit timeouts. In WTCP, the receiver performs the rate control mechanisms and computes the transmission rate for the sender; the sender transmits its current inter-packet separation with each packet. The receiver uses this information and its local state to update the transmission rate. The sender must receive ACKs, which carries both the reliability and transmission control information, periodically in order to react to the new transmission rate and perform flow control. WTCP uses selective acknowledgements scheme and does not use retransmission timers for loss recovery. It tries to remain in congestion avoidance phase at all times by detecting and reacting to incipient congestion. This makes the WTCP be more resilient to non-congestion related packet losses, thereby improving the performance over wireless links.

3.4.6 TCP EIFFEL

The Eifel described in [R. Ludwig & R. H. Katz] [RFC 3522, 2003] eliminates the retransmission ambiguity, thereby solving the problems caused by spurious timeouts and spurious fast retransmits. It allows the sender to detect whether an already initiated error recovery mechanism is in fact necessary or not by monitoring the first ACK that covers previously unacknowledged data. The sender uses the timestamp option to determine this is an acknowledgement of the original segment or of the retransmitted segment. If this ACK is for the original segment, the sender considers the retransmission is spurious and it does not have to reduce the transmission rate. The original segment is not lost due to congestion, therefore it should been delayed before it arrived at the receiver.

3.4.7 TCP REAL

TCP Real [C. Zhang & V. Tsaoussidis, 2001] is a rate based scheme extending the TCP Reno. In TCP Real, the receiver controls the sender transmission rate. The receiver uses changes in the rate of incoming segments to compute the CWND and then includes this estimate with acknowledgement that goes back to the sender. TCP Real takes the data-receiving rate as a metric to predict the network conditions. Decrease in the rate of incoming segments indicates that there is an increase in the network load therefore the CWND should be reduced. After a segment loss, the CWND is adjusted to the network conditions sooner than in the standard TCP, since the estimate of the CWND is included in the ACK. These modifications constitute the foundation for an efficient recovery strategy over heterogeneous environments with wire-line or wireless networks.

3.4.8 CONCLUSION

The end-to-end proposals described above are based on various ideas. the SMART handles multiple losses within a window of data, which is highly likely in the wireless environments, but invoke congestion control with the assumption that all losses are due to congestion. The rate based proposal try to avoid congestion and to recover quickly from random errors. Freeze TCP prevents additional data loss by making a pause in the data transfer during disconnections or handoffs. The TCP Eifel, on the other hand, limits performance degradation when delay of a segment is misinterpreted as a sign of congestion. The advantage of the end-to-end proposals are the preserve the end-to-end semantics of TCP and do not support from any intermediate nodes, thereby no additional processing is required in the network.

3.5 SUMMARY OF ABOVE PROPOSED OPTIMIZATIONS

Optimization proposals presented above use different types of approaches to improve TCP performance over wireless networks, and are categorized into four groups; split-connection, link-layer, explicit notification and end-to-end protocols.

Split-connections protocols manage to completely hide the wireless link from the wire-line part of the network by terminating the TCP connection at the base station and establishing another connection from the base station to mobile host. The transport protocol used in the latter connection can be a standard TCP or any other protocols that suit for the wireless environment, such as selective repeat protocol. The major advantage of split-connection protocols is that they provide backward compatibility with the existing protocols, thus they do not require any modification at the fixed hosts. However, they violate the end-to-end semantics of TCP and need to translate from one protocol to the other at the base station, leading significant overhead to the base station.

The link layer approaches use link layer retransmissions to improve the performance of TCP. They operate independently of higher layer protocols and try to make the wireless link appear as higher quality link, but reduced effective bandwidth. They rely on determined network elements, which collaborate at link level in order to reduce the effects of wireless link. The requirements on the link layer service may vary depending on the application. For example, the radio link protocol in the UMTS network allows many configuration parameters, such as the maximum number of retransmissions to be set. The main advantage of link layer proposals is that no modifications are required in the end points and preserve the end-to-end semantics of TCP. However, link layer

protocols could adversely affect the TCP performance [Balakrishnan, Padmanabhan, Seshan, & Katz, 1997]. Link layer protocols indeed improve the wireless link performance but fail to synchronize with the TCP sender that may help to improve the TCP performance.

Most explicit notification proposals require TCP awareness of the intermediate nodes that are responsible for transmitting explicit notifications. The sender can distinguish congestion from data loss due to wireless errors since it receives information about the transmission status. Based on this, the sender can make more informed decision. Thus, if a segment is lost for reasons other than congestion, then the sender can take appropriate actions. The main drawback of these schemes is that the end points need modifications to handle the explicit notification signal.

End-to-end proposals are based on the idea that complexity belongs in the end hosts rather than in the network. The end points are responsible for performing the necessary changes to ensure a good adaptation and do not need any support from the intermediate nodes. The main advantage of these schemes is that they can be used in any situation.

3.6 OUR PROPOSALS TO IMPROVE TCP PERFORMANCE OVER WIRELESS LINK

We have carefully analyzed the extensive literature on TCP optimizations over wireless networks. They all have the same goal to improve the TCP performance over wireless networks, but use different approaches to achieve the goal. The efficiency of those optimizations is network dependent. Based on the findings and considering the inefficiency of the TCP over wireless link, we propose some new schemes that optimize the TCP performance. We consider the following optimizations in our proposed schemes.

- Distinguish congestion losses from non-congestion losses to fine tune the TCP to perform well in both wired and wireless environment
- Use link layer approach without local retransmissions (retransmissions of cached packets by the proxy) to minimize the base station overhead
- Minimize the intermediate node dependencies
- Early timeout detection to speed up the recovery process
- Analyzing the impact of packet loss pattern on the performance of TCP

- Using dynamic congestion window size based in loss probability
- Design an analytical model with enhanced recovery mechanism
- Wireless timeout detection to minimize spurious timeouts

Since it is very difficult to cover the entire area of congestion avoidance and control issues and due to the time constraint, our research scope is restricted to the improvement of TCP performances over wireless networks without countering hand-offs effects. We have implemented our proposed schemes in the WLAN and UMTS networks and demonstrated their efficacy in the performance improvement of TCP in the following Chapters.

Chapter IV.

RNF Scheme for TCP Enhancement over a 802.11 Wireless LAN

In this Chapter, we propose a wireless loss detection technique that allows the TCP sender to distinguish wireless packet losses from the congestion related packet losses. This scheme employs an enhancement proxy, called wireless loss detector (WLD), in the base station that monitors the TCP flow between the base station and the mobile host in both directions, and detects the wireless packet loss. When a wireless packet loss is detected, it sends a radio network feedback (RNF) with the acknowledgements that arrive from the mobile host. One of the TCP header reserved bits is utilized to carry the RNF and a minimal change to the standard TCP congestion control mechanism is required to accommodate this effect.

Section 4.1 gives an overview of wireless LAN (WLAN) technology and motivates the need for the RNF mechanism. The concept behind the RNF mechanism and the methodology applied in developing the RNF mechanism are explained in Section 4.2. The implementation details of the RNF mechanism in a WLAN environment is outlined in Section 4.3. The performance of our proposed scheme is explained and compared with that of the standard WLAN and the WLAN with Snoop enhancing proxy in Section 4.4. At last, we draw our conclusions and present the guidelines for further improvement.

4.1 OVERVIEW OF WIRELESS LOCAL AREA NETWORK

Wireless LANs are now one of the most important access network technologies in the Internet today. Although many technologies and standards for wireless LANs were developed in the 1990s, the IEEE 802.11 WLAN, also known as Wi-Fi, has emerged as the initial standard for WLANs [Kurose & Ross, 2003]. There are several 802.11 standards for WLANs technology, including 802.11a, 802.11b, and 820.11g. Main characteristics of these standards are summarized in Table 4.1. The 802.11 standards share many characteristics; they all use the same medium access protocol, use the same frame structure for their link layer frames, have the ability to

support multiple transmission modes in order to reach out over greater distances and allow for both *infrastructure mode* and *ad hoc mode*. However, as can be from Table 4.1, 802.11 standards have some major differences at the physical layer.

Standard	Frequency Range	Data Rate
802.11a	5.1 – 5.8 GHz	Up to 54 Mbps
802.11b	2.4 – 2.485 GHz	Up to 11 Mbps
802.11g	2.4 – 2.485 GHz	Up to 54 Mbps
802.11n	2.4 – 5 GHz	Up to 248 Mbps

Table 4-1 Summary of IEEE 802.11 Standards

The 802.11a WLANs operate in the 5 GHz frequency range and can offer transmission rate up to 54 Mbps, but they have a shorter transmission distance for a given power level and suffer more from multi-path fading. The 802.11b WLANs have data rate of 11 Mbps and operate in the unlicensed frequency bands of 2.4 – 2.485 MHz, competing for frequency spectrum with 2.4 MHz phones and microwave ovens. 802.11g WLANs operate in the same lower frequency band as 802.11b, but with the higher transmission rate of 802.11a. They employ orthogonal frequency division multiplexing (OFDM), the modulation scheme used in 802.11a WLAN, to obtain higher data rate and can fall back to speeds of 6 Mbps. This feature makes 802.11b and 802.11g WLAN devices compatible within a single network. The 802.11n is a proposed amendment, which improves upon the previous standards by adding multiple-input multiple-output (MIMO) and many other new features.

4.1.1 THE 802.11 ARCHITECTURE

Figure 4.1 illustrates the principal components of the 802.11 WLAN architecture. An 802.11 WLAN is based on a cellular architecture where the system is subdivided into cells. Each cell, called Basic Service Set (BSS), is controlled by a Base Station called Access Point (AP). When a WLAN is formed from several BSSs, APs are connected through a backbone network, typically Ethernet, called a Distribution System (DS). The whole interconnected WLAN including the different BSSs with their respective APs and the DS is called an Extended Service Set (ESS). Wireless LANs that deploy APs are often referred to as infrastructure wireless LANs.

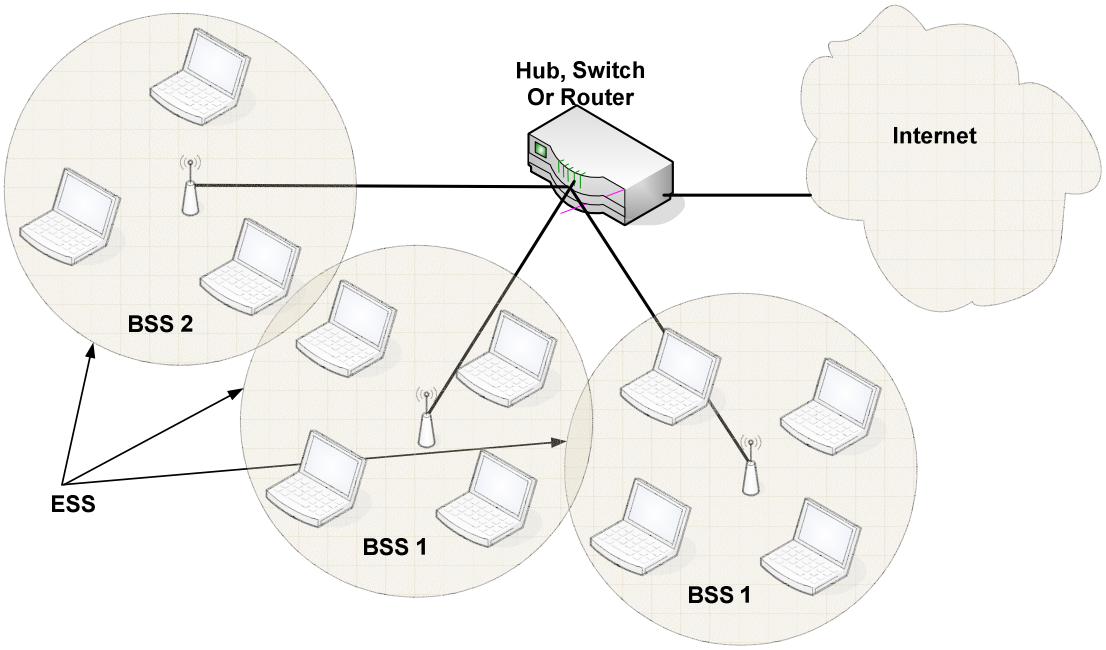

Figure 4-1 IEEE 802.11 WLAN Architecture

Figure 4.2 shows that 802.11 stations can also group themselves together to form an *ad hoc* network; a network with no central control and with no connection to the Internet. In *ad hoc* networks, several wireless stations join together to establish a peer-to-peer communication. Each client communicates directly with the other clients within the network. *Ad hoc* mode is designed such that only the clients within the transmission range of each other can communicate. If a client in an *ad hoc* network wishes to communicate outside the cell, a member of the cell must operate as a gateway and perform routing. They typically require no administration and share the network resources without a central server. There has been tremendous interests in ad hoc networking as communicating portable devices continue to grow. However, we will focus our studies on the infrastructure wireless LANs.

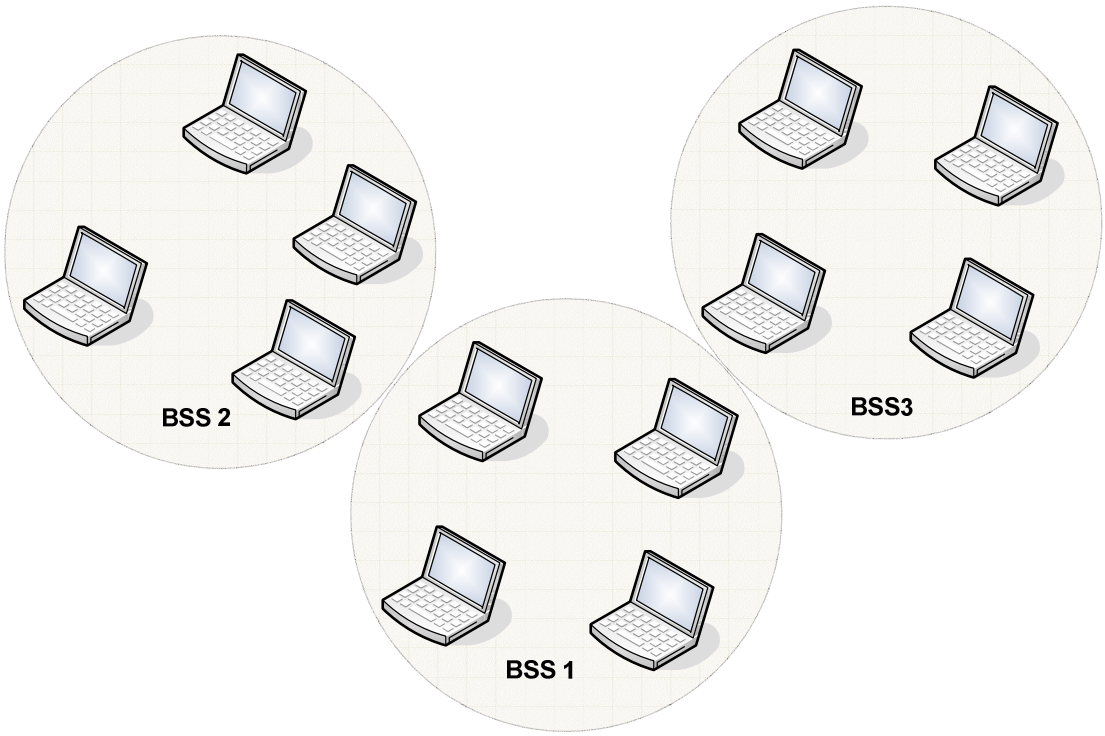

Figure 4-2 An IEEE 802.11 ad hoc network

4.1.2 THE 802.11 MAC PROTOCOL

The 802.11 standard specifies the MAC and Physical layers that provide a variety of functions to support the operation of 802.11 wireless LANs. The MAC layer manages and maintains communications between 802.11 stations by coordinating access to a shared radio channel. As illustrated in Figure 4.3, the 802.11 physical layer has Frequency Hopping (FH) Spread Spectrum, Direct Sequence (DS) Spread Spectrum and the Infrared (IR). The MAC interacts with the physical layer by passing the MAC frame and receiving the MAC frame.

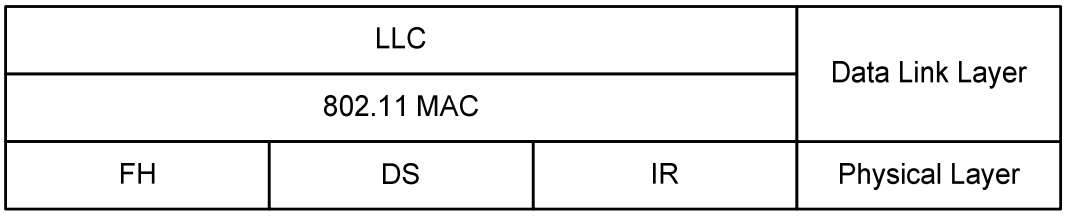

Figure 4-3 IEEE 802.11 MAC Layer

Besides the standard MAC layer functionality, the 802.11 MAC performs other functions, such as packet fragmentation, retransmissions and acknowledgements. The MAC provides access to the medium through coordinated functions (CFs) to support both asynchronous and time-

bound traffic; the Distributed Coordination Function (DCF) and the Point Coordination Function (PCF).

The DCF is basically a Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) mechanism and provides contention-based access. In CSMA/CA, a wireless station that wants to transmit performs the following sequence as illustrated in Figure 4.4.

1. Senses the medium. If the medium is idle for a specified time called Distributed Inter Frame Space (DIFS) then the station moves into a contention window and starts a random back off timer when waiting for the contention window.

2. If it is the first station to finish its allocated number of slot times, it transmits (RTS) signal, otherwise it stops counting down their back off timer until the medium becomes idle again.

3. The target receiver replies with clear to send (CTS) signal after inter frame space (IFS) interval, which is the time interval between transmissions of frames.

4. On receiving CTS, the sender can transmit the data frame. All other stations set the network allocation vector (NAV) value. Each station calculates the amount of time required to send the frame based on the length of the frame and the data rate and places this value in the duration field of the MAC header, which is then used by the other station as a basis for setting their NAV value.

5. If the receiver gets the frame correctly, it sends an ACK to the sender after SIFS interval.

6. If the sender does not receive the ACK, it will keep transmitting the packet after randomly backing off until it either gets acknowledged or is discarded after a certain number of retransmissions.

7. Any other station that attempts to the medium waits for DIFS and a back-off procedure is invoked.

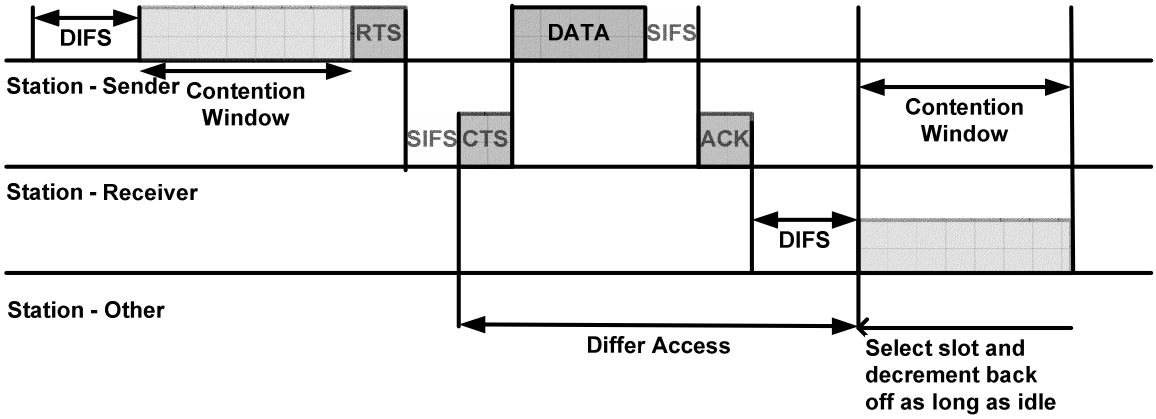

Figure 4-4 CSMA/CA mechanism

An important aspect of the DCF is a random back off timer. If the medium is busy the station must wait for DIFS interval plus a random number of slot times. The time between the end of DIFS and the beginning of the next frame is known as the contention window. Each station starts a random back off timer when waiting for the contention window. The first station to finish its allocated number of slot times begins transmission and the other stations stop counting down their back off timer until the medium becomes idle again. The random delay causes stations to wait for different period of times and avoid all of them sensing the medium at exactly the same time. This in turn reduces the number of collisions and corresponding retransmissions significantly.

The 802.11 MAC protocol also includes a scheme using RTS/CTS control frames that helps avoid collisions even in the presence of hidden terminals, which are defined as the terminals beyond the communication range of the transmitter but within that of the receiver. Although the RTS/CTS exchange can help collisions, it also introduces delay and consumes channel resources. For this reason, the RTS/CTS exchange is only used to reserve the channel for the transmission of a long data frame. In practice, each wireless channel can set RTS threshold such that the RTS/CTS sequence is used only when the frame is longer than threshold. For many wireless stations, the default RTS threshold value is larger than the maximum frame length, so the RTS/CTS sequence is skipped for all data frames sent [Kurose & Ross, 2005].

PCF provides contention free access and service for supporting time sensitive data via a totally centralized polling mechanism. In PCF, APs send beacon frames at regular intervals. Between these beacons frames, PCF defines two periods; the contention free period (CFP) and contention

period (CP). In CP, the DCF is simply used. In CFP, the AP sends contention free beacon frame to each station in the BSS after it confirms that the medium is idle for point inter frame space (PIFS). Beacon frame contains the information on the maximum duration of the CFP, beacon interval, and BSS identifier. All stations in BSS set their NAV value and not to send any packet in the CFP after receiving a beacon.

4.1.3 THE IEEE 802.11 FRAME

The 802.11 frame shown in Figure 4.5 contains a number of fields that are specific to its use for wireless links. The payload field can hold data up to 2312 bytes. However, it is typically fewer than 1500 bytes, holding IP datagram or an ARP packet. As with an Ethernet frame, an 802.11 frame includes a cyclic redundancy check (CRC) so that the receiver can detect any bit errors in the received frame. The CRC is more useful in wireless LANs, since it has very high bit errors. There are four address fields, each of which can hold up to 6 bytes MAC address.

- Address 1 is the MAC address of the station that transmits the frame.
- Address 2 is the MAC address of the station that is to receive the frame.
- Address 3 is the MAC address of the router interface to which BSSs are connected.
- Address 4 is only used in ad hoc networks.

The sender may send multiple copies of a given frame if the acknowledgement gets lost. The use of sequence number field allows the receiver to distinguish between a newly transmitted frame and the retransmission of previous frame. The 802.11 protocol allows a transmitting station to reserve the channel for period of time required to send the frame, including the time to transmit its data frame and the time to transmit the acknowledgement. This duration value is placed in the duration field.

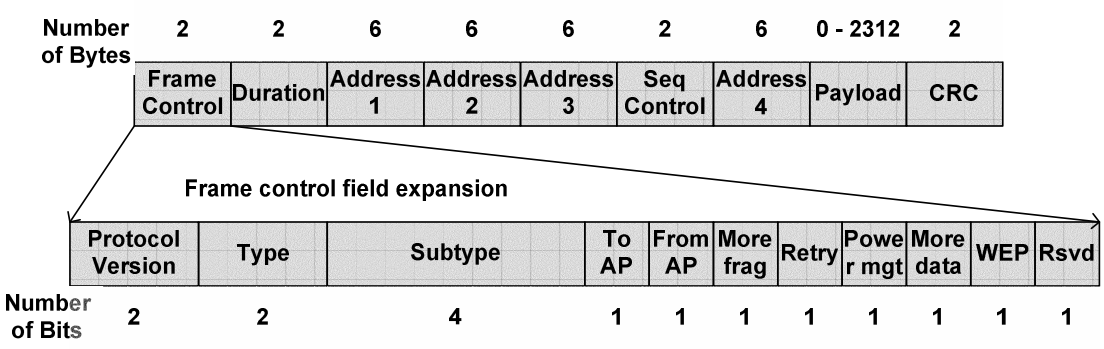

Figure 4-5 The 802.11 Frame

The control frame assists in the delivery of data frames between stations and includes many subfields. The type and subtype fields are used to distinguish the association RTS, CTS, ACK, and data frames. The *To AP and From AP* frame are used to define the different address fields. More fragments field is used to indicate that more fragments belonging to the same frame following this current fragment. Retry field indicates that this fragment is a retransmission of a previously transmitted fragment. This will be used by the receiver station to recognize duplicate transmission of frames. Finally, the wired equivalent privacy (WEP) field indicates whether encryption is enabled or not. More complete description of MAC frame can be found in [O'Hara & Petrick].

4.2 THE RADIO NETWORK FEEDBACK MECHANISM

One of the most challenging and interesting trends in recent computer networks is the integration of mobile communications. Wireless LANs based on the IEEE 802.11 MAC protocol [O'Hara & Petrick] are becoming ubiquitous as they can deliver services commonly found in wired networks. However, the performance of transport layer protocols, such as TCP, over 802.11 may be degraded considerably due to the characteristics of the wireless medium that suffers from significantly high bit error rates. In this Chapter, we present a technique that detects wireless packet losses and enables the TCP sender to completely distinguish wireless packet losses from congestion related losses. It also helps the TCP sender to minimize the unnecessary TCP timeouts, which leads to a reduction in CWND and unnecessary retransmissions of packets using Go-Back-N mechanism, when non-congestion related losses occur.

4.2.1 PROPOSED RNF SCHEME

In previously proposed schemes in the literature, TCP performance has been improved essentially by enhancing link layer protocols. We adopt the alternative view that TCP

improvements should be achieved by tuning TCP itself to utilize the available network resources efficiently in both wire-line and wireless environments. Our proposed scheme introduces a proxy called wireless loss detector (WLD) that monitors the WLAN network radio interface and, with the aid of one of the TCP header reserved bits, notifies the TCP sender of packet losses across the wireless link. The TCP End-to-End semantic is maintained but it is modified in order to adapt to the characteristics of the wireless environment.

WLD enhancement proxy is introduced between the 802.11 WLAN MAC and the IP layers of 802.11 WLAN base stations. It monitors the TCP flow between the mobile hosts and the base station. On receiving data packets from the IP layer that are destined for mobile hosts, WLD proxy obtains the TCP header information from the IP datagram, assuming it is not encrypted, and maintains them in a Cache Table. It also keeps a Connection Table in order to be able to support multiple TCP connections. Figure 4.6 shows the flow control to extract the TCP header information from IP datagram packets. It should also be noted that RNF scheme cannot distinguish a wireless packet loss due to contention or due to channel BER. However, RNF can be further modified to incorporate this. We leave this a future work.

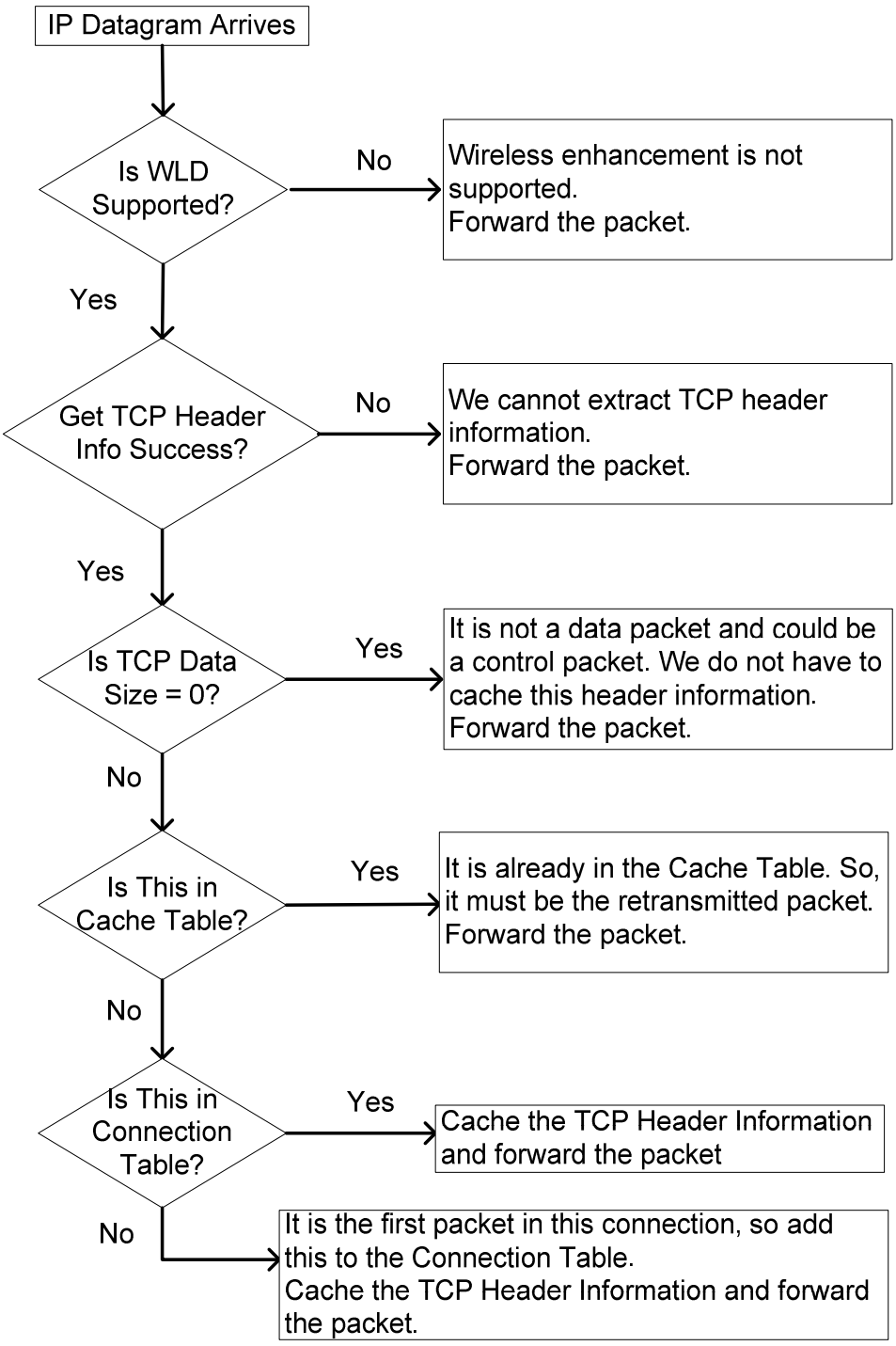

Figure 4-6 WLD flow control for caching TCP header information

On the arrival of ACK packets from the MAC layer, WLD proxy uses both the Cache Table and Connection Table to detect wireless packet losses. If a wireless packet loss is detected, it notifies the TCP sender of this effect by utilizing the control bit next to the CWR flag in the reserved field of the TCP header, called Radio Network Feedback (RNF) flag, as shown in Figure

4.7. It should be noted that the maximum header length is 60 bytes. TCP header length is usually 20 bytes, but can have up to 40 bytes for options. Since we only cache the TCP header information, which is of 20 bytes long, our proposed scheme does not add much overhead to the base station.

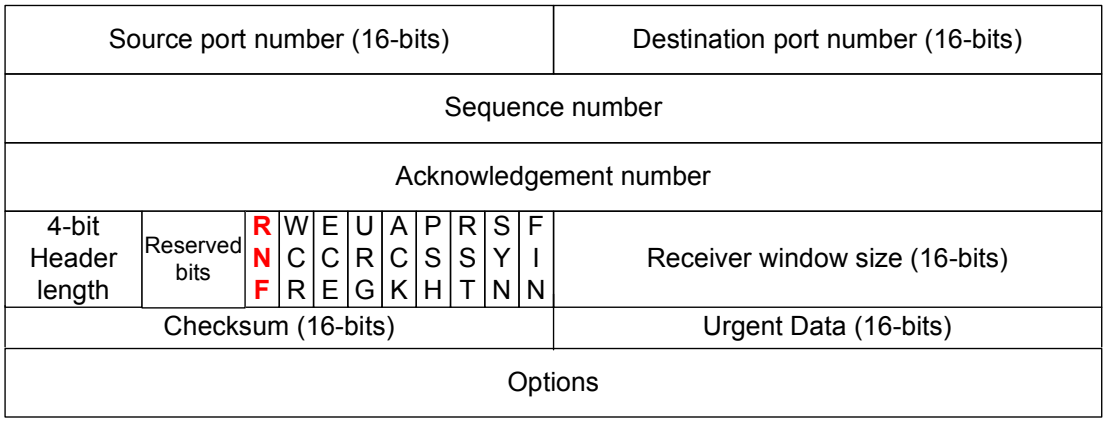

Figure 4-7 TCP header with RNF flag

Flow control for the wireless packet loss detection is shown in Figure 4.8. The main contribution of this scheme is the wireless loss detection, which is the key to successful TCP performance improvement, and the TCP modification to accommodate this effect. Once the TCP sender can completely distinguish between the packet losses due to the congestion and wireless errors, TCP congestion control mechanism can be tuned to adapt to the network environments. It is critical to decide whether to reduce the CWND size when a TCP sender detects wireless packet losses. We define α to be the bandwidth (BW) utilization factor that the TCP sender uses to calculate the CWND size during the fast retransmit and recovery phase. The value of α can be between 0.5 and 1. In the standard TCP Reno and TCP New Reno, the α is assigned value of 0.5; they halve the CWND when a packet drop is detected by three duplicate ACKs. The value of α can be optimized for wireless packet loss recovery and it is discussed in detail in Section 4.3.5.

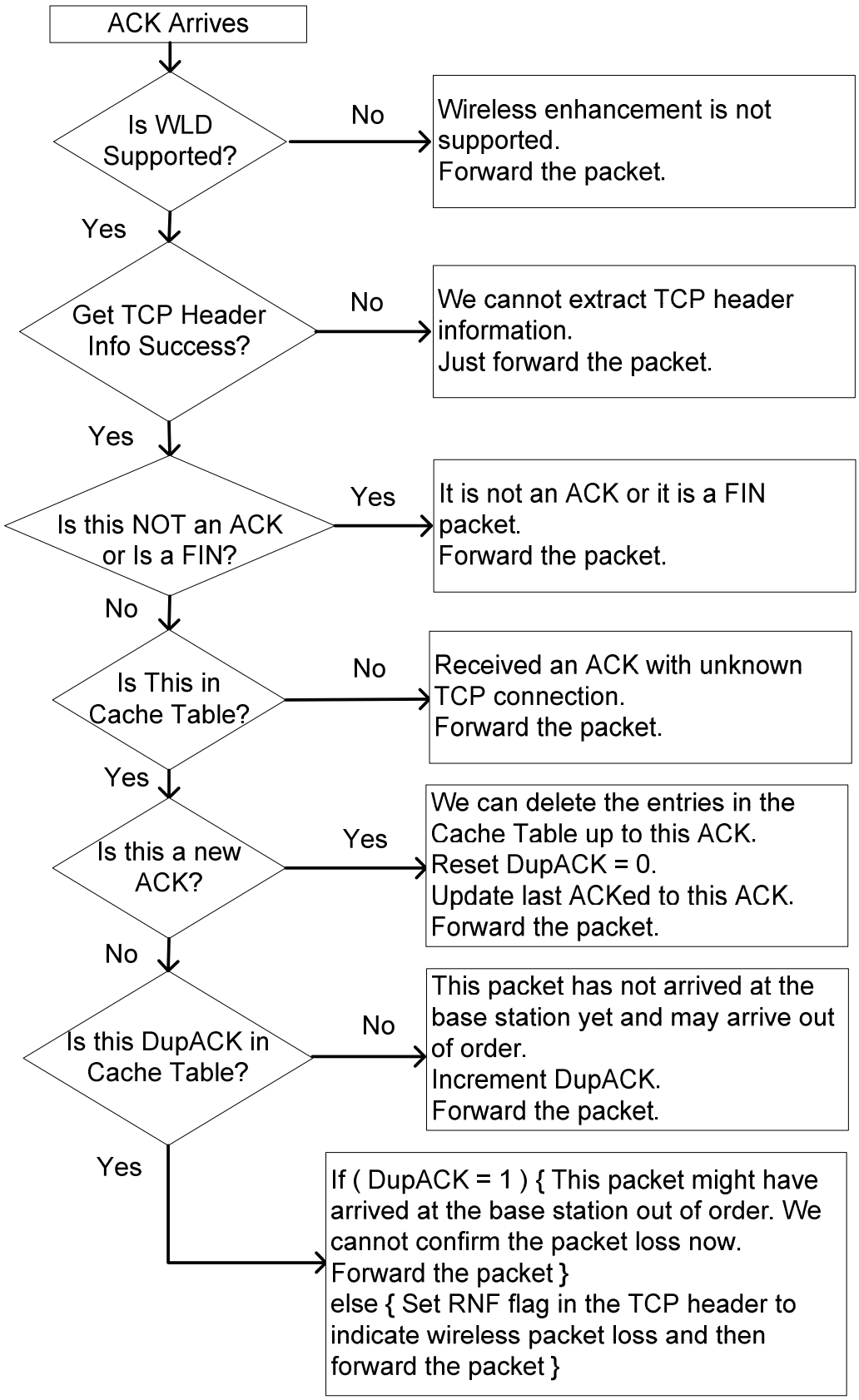

Figure 4-8Wireless loss detection flow control

4.3 THE IMPLEMENTATION DETAILS OF THE RNF MECHANISM

The implementation of the proposed scheme, and the rationale for it, are briefly explained. The scheme consists of two parts; wireless loss detection and fine-tuning of the TCP congestion control mechanism to adapt to the network condition.

4.3.1 WIRELESS LOSS DETECTION

WLD proxy maintains three data structures to handle the wireless loss detection; the TCP Information Structure, the TCP Connection Table and the Global Structure for Cache Table, as shown in Figures 4.9, 4.10 and 4.11 respectively.

```
typedef struct
{
      unsigned  int   src_ip;
      unsigned  int   dest_ip;
      int             src_port;
      int             dest_port;
      unsigned  int   seq_num;
      unsigned  int   ack_num;
      unsigned  int   rcv_win;
      int             urgent_pointer;
      int             data_len;
      TcpT_Flag       flags;
} TcpInfo;
```

Figure 4-9 TCP Information Data Structure

```
typedef struct
{
      unsigned   int    src_ip;
      unsigned   int    dest_ip;
      int               src_port;
      int               dest_port;
      unsigned   int    last_seq_num;
      unsigned   int    last_ack_num;
      int               repeat_ack;
      int               fin_flag;
      unsigned          fin_seq_num;
} struct_TcpConnectionTable;
```

Figure 4-10 TCP Connection Table Data Structure

```
typedef struct
{
      TcpInfo *tcp_infoPt[WLD_CACHE_SIZE];
      int ici_addr[WLD_CACHE_SIZE];
      int max_cached;   /* Maximum number of packet header that can be cached at
                           the same time */
      int num_cached;   /* Total number of packets header that has been cached */
      int num_removed;   /* Total number of packets header that has been removed */
      int curr_cached;  /* Current number of packets header cached */
} struct_TcpHeaderCache;
```

Figure 4-11 Global Structure for Cache Table

WLD proxy maintains five states, as shown in Figure 4.12, depending on the transition variables, and its functionality is explained as follows.

- INIT state initializes both the Cache Table and the TCP Connection Table with the necessary information for each TCP connection passing through the base station.

- WAIT state is the default state and depending on the transition variables, the state will change between the corresponding state and the default state.

- Monitor Data state extracts TCP header information from the IP datagram and determines the extracted TCP header is already in the Cache Table. If this TCP header is not in the Cache Table, it will add it to the Cache Table after confirming that this connection information exists in the TCP Connection Table. Otherwise, it will first add the connection information to the TCP Connection Table and then cache the TCP header information to the Cache Table.

- Monitor ACK state extracts TCP header information from the IP datagram, coming form the mobile receiver, and determines this is an ACK packet before triggering its WLD algorithm to detect wireless packet loss. It uses the extracted TCP header information, the Cache Table and the TCP Connection Table to confirm a wireless packet loss, as illustrated in the Wireless loss detection flow control in Figure 4.8. If a wireless packet loss is detected, it will set the RNF flag of the ACK packet and forward it to the TCP sender. Otherwise it will forward the packet as it enters. On receiving a new ACK, it will destroy the entries in the Cache Table up to this ACK in order to further reduce the overload at the base station. Finally, it will destroy entries both in the Cache Table and the Connection Table on receiving the ACK packet with FIN bit set.

- Timeout state maintains a timer that can be used to make the TCP sender aware of delay spikes, defined as sudden large delays, across the wireless link that causes spurious TCP timeouts. TCP can be further enhanced to distinguish spurious timeouts from the normal timeouts. It should be noted that Monitor DATA state and the Monitor ACK state can be utilized to record the arrival times of data packets and their corresponding ACKs at the base station. Timeout state can then use this information to calculate the wireless RTT (W-RTT) and the wireless RTO (W-RTO) and to detect delay spikes. This feature is not considered in this proposal and is left for our future work.

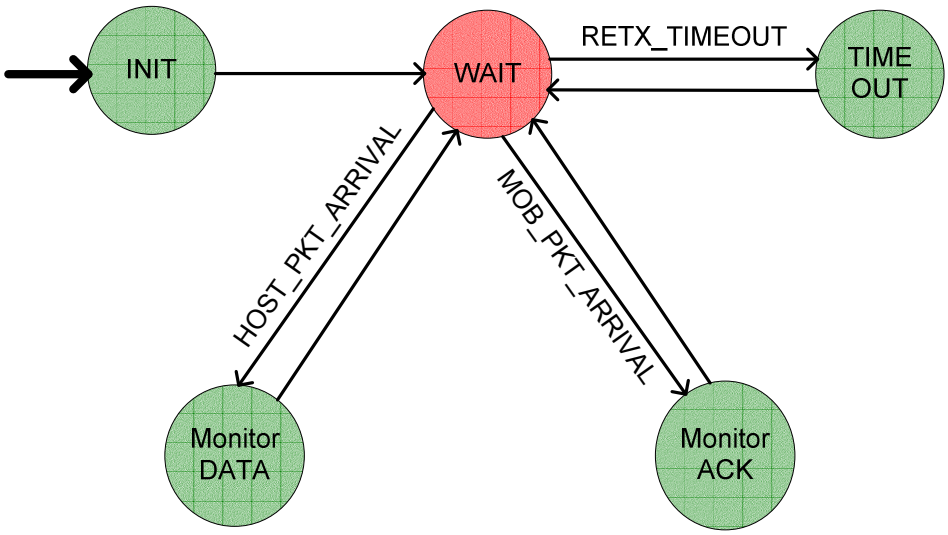

Figure 4-12 WLD Process Model

4.3.2 TCP CONGESTION CONTROL MODIFICATION

The standard TCP congestion control mechanisms are tuned to perform well in traditional wire-line networks, where packet losses occur mostly because of congestion. However, networks with wireless links suffer form significant losses due to high bit errors and handoffs. TCP responds to all losses by invoking congestion control and avoidance algorithms, resulting in degraded End-to-End performance in wireless environment. The TCP congestion control should be modified to utilize the available bandwidth efficiently in wireless environments. WLD proxy enables the TCP sender to confirm a packet loss due to wireless error when it receives an acknowledgement with the RNF flag set. The RNF flag is added to the TCP header as;

```
#define TCPC_FLAG_RNF  0x100
```

In standard TCP implementations, the TCP sender concludes that the network has dropped a packet when it receives three duplicate ACKs. It then immediately retransmits the dropped packet without waiting for the retransmission timer to expire. The rationale is that the sooner the fast retransmit occurs, the better TCP performs because it avoids unnecessary TCP timeouts. With this in mind, and the fact that TCP sender can confirm any wireless packet drops when it receives duplicate ACKs with the RNF flag set, it is fine tuned to retransmit the packets dropped across the wireless link when it receives two duplicate ACKs with the RNF flag set. It should be noticed that the TCP sender, considering the delayed packet arrivals at the base station, still waits

for two duplicate ACKs with RNF flag set even though it has received confirmation that the packet has been dropped across the wireless link.

Now it comes to deciding whether to reduce the CWND when a TCP sender detects a wireless packet drop. It is very important to keep the existing implementation of TCP while modifying it to accommodate new features. We have added two new features to the standard TCP without making any changes to its current implementation; one is to enable the TCP sender to process the fast retransmit and fast recovery mechanism depending on the cause of a packet loss. The other implements wireless fast retransmit and fast recovery mechanism to quickly recover form wireless related losses with the use of the BW utilization factor, which will have a great impact on the TCP performance improvement over wireless links.

For example, if BW utilization factor is assigned the value of 1, it will improve the TCP throughput performance by CWND/2 per RTT after each wireless packet recovery given that the receiver window is bigger than the CWND. It will also increase the chance of recovering multiple packet losses within a window of data, which are highly likely in wireless environments, thereby reducing unnecessary TCP retransmission timeouts. However, this may have some adverse effect on networks in case of heavy loss across the wireless links. The BW utilization factor can be optimized to perform well over network with wireless links. We leave this phenomenon for our future work and with this scheme, we implement the wireless fast retransmit and fast recovery mechanism with the BW utilization factor of 0.75 and 1.

Figure 4.13 shows the flow control for the TCP ACK processing, which distinguishes between packet losses due to congestion and packet losses due to wireless error with the aid of RNF flag. It triggers the standard fast retransmit and fast recovery processing if the loss is due to congestion, otherwise call the wireless fast retransmit and fast recovery processing, which is added to the standard TCP as part of the TCP modification to enhance its performance over the wireless networks.

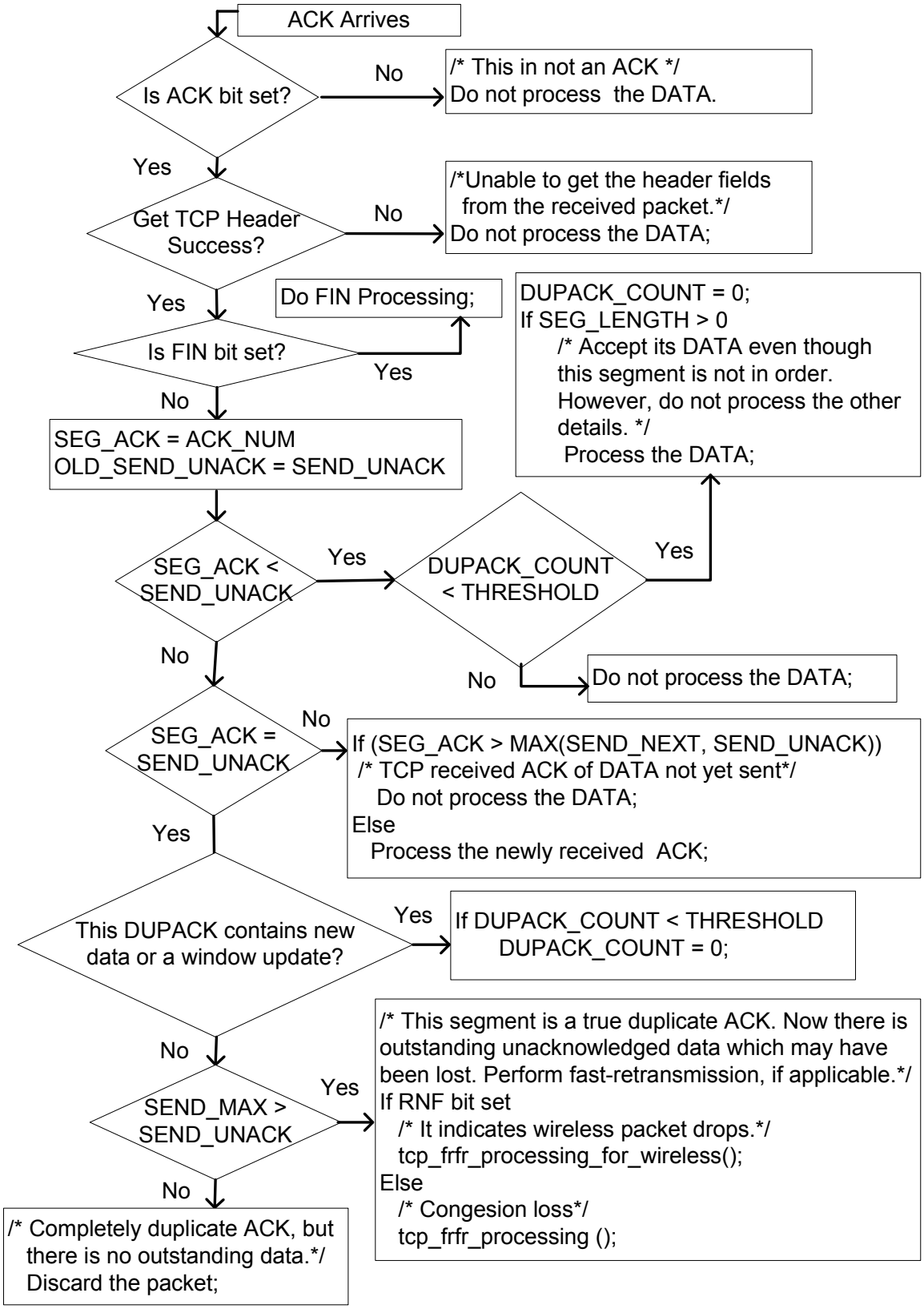

Figure 4-13 ACK processing with RNF flag

4.3.2.1 The Wireless Fast Retransmit and Fast Recovery Processing

The wireless fast retransmit and fast recovery processing triggers wireless fast retransmit algorithm when it receives two duplicate ACKs with RNF flag set. Similar to the standard TCP fast retransmit algorithm, it considers the flight size, which is the number of unacknowledged packets in the network, in calculating the *SSTHRESH* value. On fast retransmitting the lost packet, the CWND is set to *SSTHRESH* times the BW utilization factor. The number of new packets can be transmitted during the fast recovery process depend on the CWND value. If BW utilization factor is assigned a value of 0.5, there can be maximum of $(CWND/2 - P_D)$ number of packets transmitted, where P_D is the number of dropped packets within that window of data. It should be noted that it will take the TCP sender *CWND*/2 of RTT period to utilize the available BW in which it was operating prior to a single packet drop experienced. If the packet drop were due to wireless error, TCP sender under-utilizes the available BW, causing performance degradation.

4.4 PERFORMANCE EVALUATION OF OUR PROPOSED SCHEME BASED ON OPNET SIMULATION

The OPNET [OPNET Technologies Inc] simulation tool is used throughout our studies to implement and evaluate the proposed scheme based on simulation studies. We have implemented the WLD scheme in both an 802.11 WLAN Server and an 802.11 WLAN Router and Extensive simulations were run to get the mean TCP throughput with less than 5% error margin. Modified TCP Reno with the default parameters is used in all simulation scenarios. Selected TCP Reno and WLAN parameter values [OPNET Technologies Inc]are given in Tables 4.2 and 4.3, respectively. The received power of a packet is inversely proportional to the distance that it travels. The lower the reception power threshold, the greater will be the transmission range of a packet since packets with lower power will be accepted by the radio receiver. The higher the transmission power, the greater the distance that the packet can be transmitted over. With 11Mbps data rate, 5mW transmission power and -95 dBm, the transmission rate can be up to 1094m [OPNET Technologies Inc].

Maximum Segment Size (MSS)	1460 bytes
Receive Window Size at Mobile Hosts	Default
Delayed ACK Mechanism	Segment/Clock Based
Maximum ACK Delay	0.2 seconds
Maximum ACK Segments	2
Slow-Start Initial Count	1 MSS
Duplicate ACK Threshold	3
Fast Retransmit	Enabled
Fast Recovery	Reno
Selective ACK (SACK)	Disabled
Time Stamp	Disabled
Initial RTO	3.0 seconds
RTT Gain	0.125
Deviation Gain	0.25
RTT Deviation Coefficient	4.0

Table 4-2 Selected TCP Reno Parameter values

Physical Characteristics	Direct Sequence
Data Rate	11 Mbps
Transmit Power	0.005 W
Packet Reception-Power Threshold	-95 dBm
Fragmentation Threshold	None
Short Retry Limit	7
Long Retry Limit	4
Max Receive Lifetime	0.5 seconds
Buffer Size	256000 bits
PCF Parameters	Disabled
Time Stamp	Disabled
BSS Identifier	Auto Assigned

Table 4-3 Selected WLAN Parameter values

4.4.1 EXPERIMENT WITH AN 802.11 WLAN SERVER

The network model used in this study is shown in Figure 4.14. It consists of two WLAN nodes, a WLAN Server and Mobile Hosts (MHs). The two WLAN nodes are Application Configuration and the Profile Configuration nodes, which are configured to generate FTP traffic comprising a 160,000 byte file upload. The WLAN Server is implemented, in turn, with a standard WLAN, a WLAN with the Snoop proxy and a WLAN with WLD proxy to compare their relative performances. OPNET representations of a WLAN Server with the WLD proxy and a WLAN MH with wireless loss packet generator (WL-PEG) are shown in Figures 4.15 and 4.16 respectively. MHs are configured to drop 2 % of IP datagrams, using a uniform probability distribution. Modified TCP Reno with the default parameters is used in all simulation scenarios. Selected TCP Reno and WLAN parameter values are given in Tables 4.2 and 4.3, respectively.

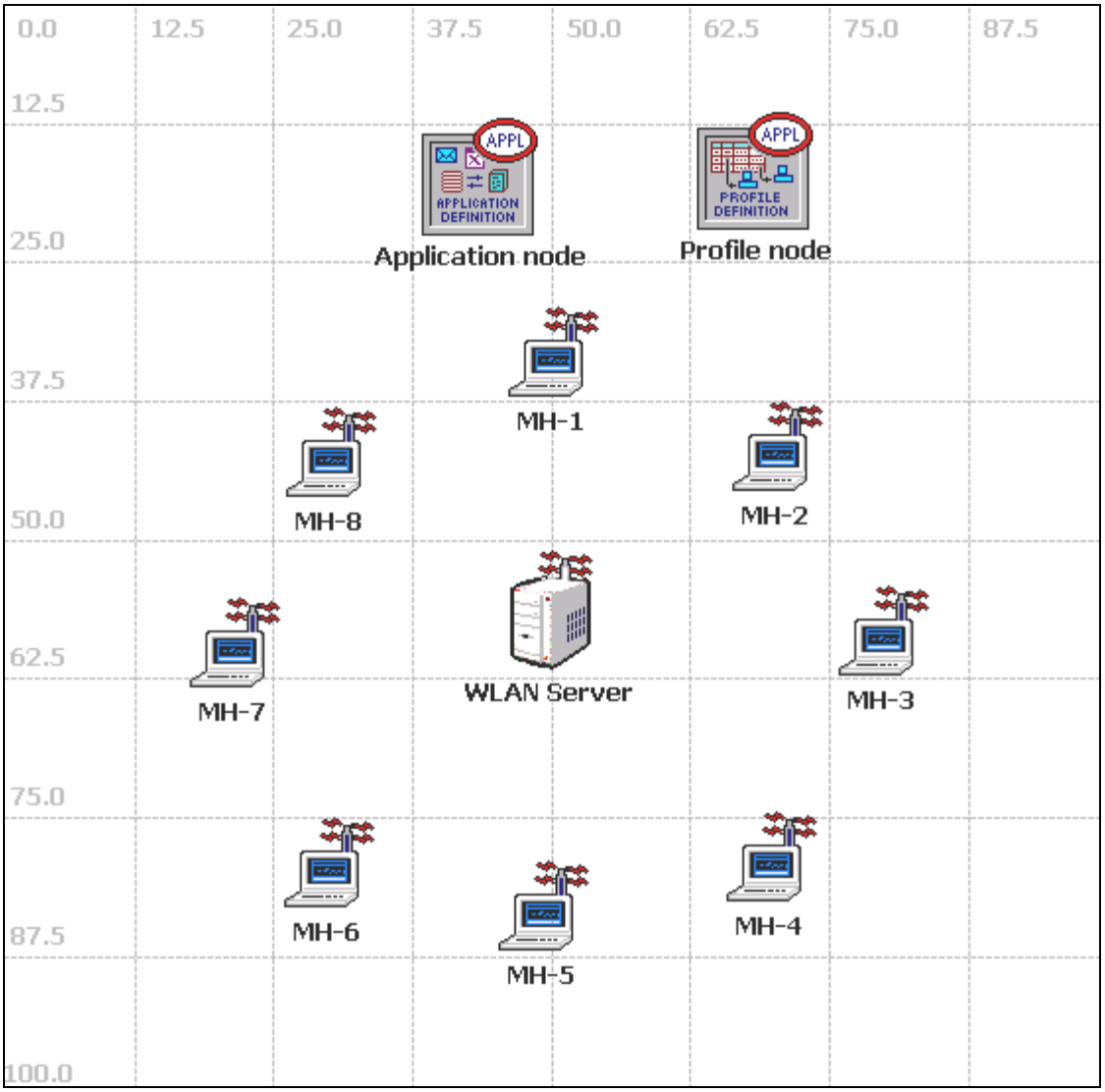

Figure 4-14 WLAN network model with an 802.11 WLAN Server

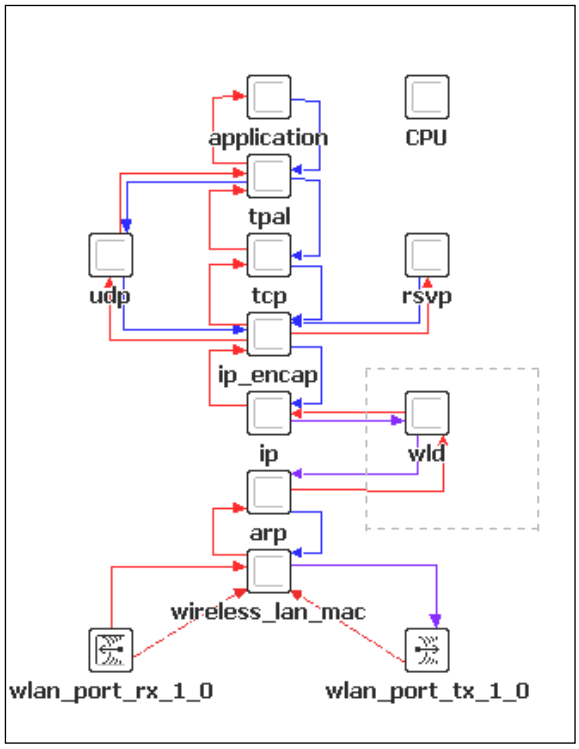

Figure 4-15 OPNET representation of a WLAN Server with WLD proxy

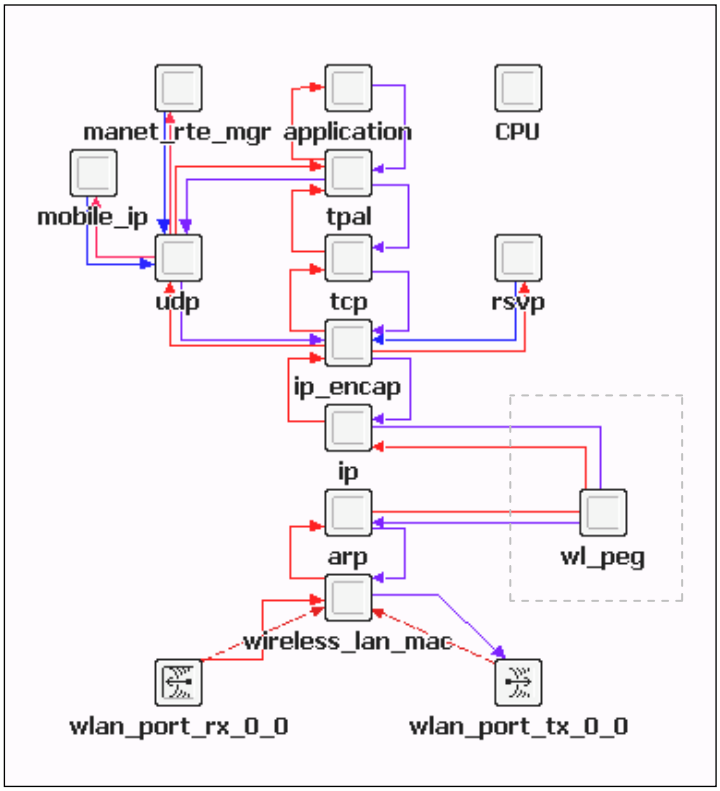

Figure 4-16 OPNET representation of a WLAN MH with WL-PEG

*4.4.1.1 **Simulation Results and Discussion***

Extensive simulations were run to compare the performance of a WLAN with the WLD enhancement proxy with a standard WLAN and a WLAN with the Snoop proxy. Figure 4.17 shows the TCP CWND size responses together with packet drops. It can be seen that the proposed scheme successfully distinguished packet losses due to wireless error from congestion and recovered from wireless packet losses without reducing the CWND while the standard WLAN, as expected, reacted to each packet loss and reduced its CWND. WLD proxy helped the TCP sender to avoid unnecessary spurious TCP timeouts. The WLAN with Snoop proxy also managed to recover from most of the wireless errors. However, it experienced timeouts due to its inability to exchange the wireless effects with the TCP sender.

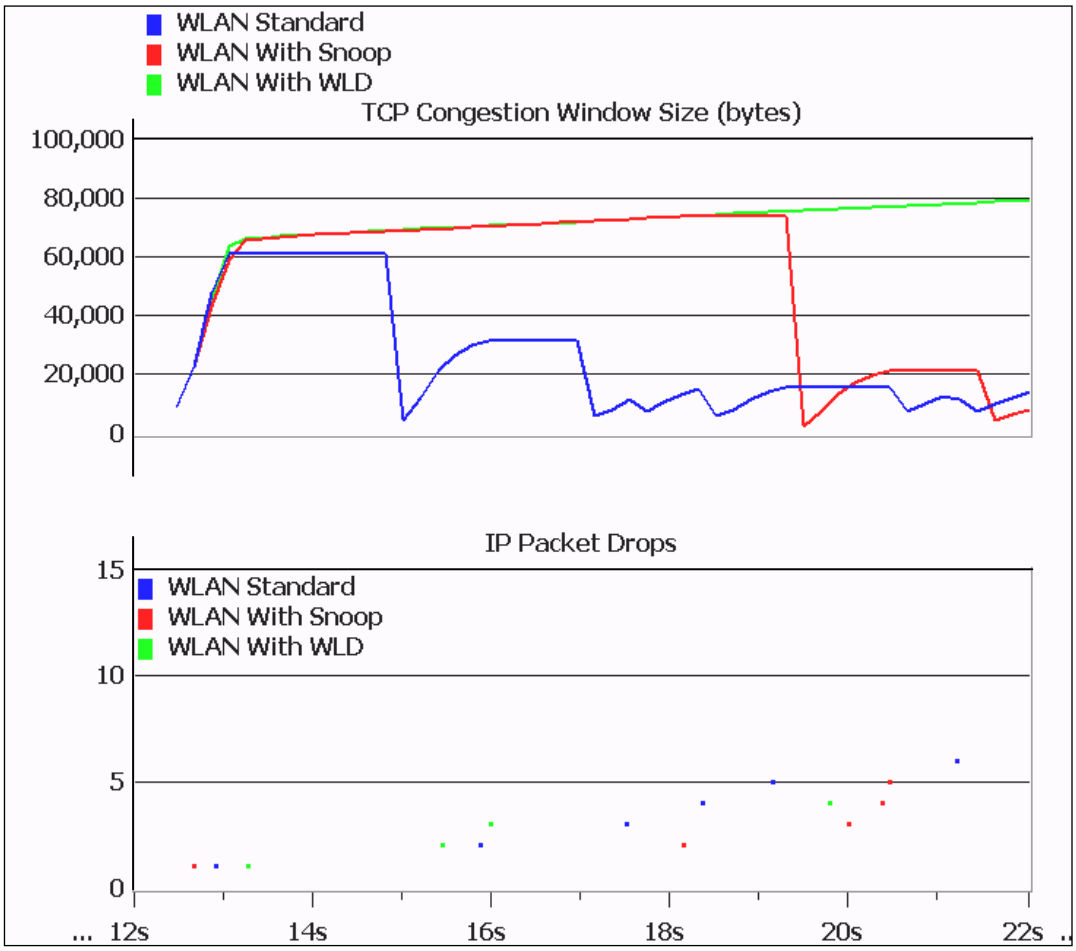

Figure 4-17 Comparison of TCP CWND responses with for MH-5

The TP timeouts makes the TCP sender perform slow-start, assuming all outstanding unacknowledged packets are lost. This effect can be observed from Figures 4.17 and 4.18 and it can be highly undesirable form user's perspective because the system may look like unstable.

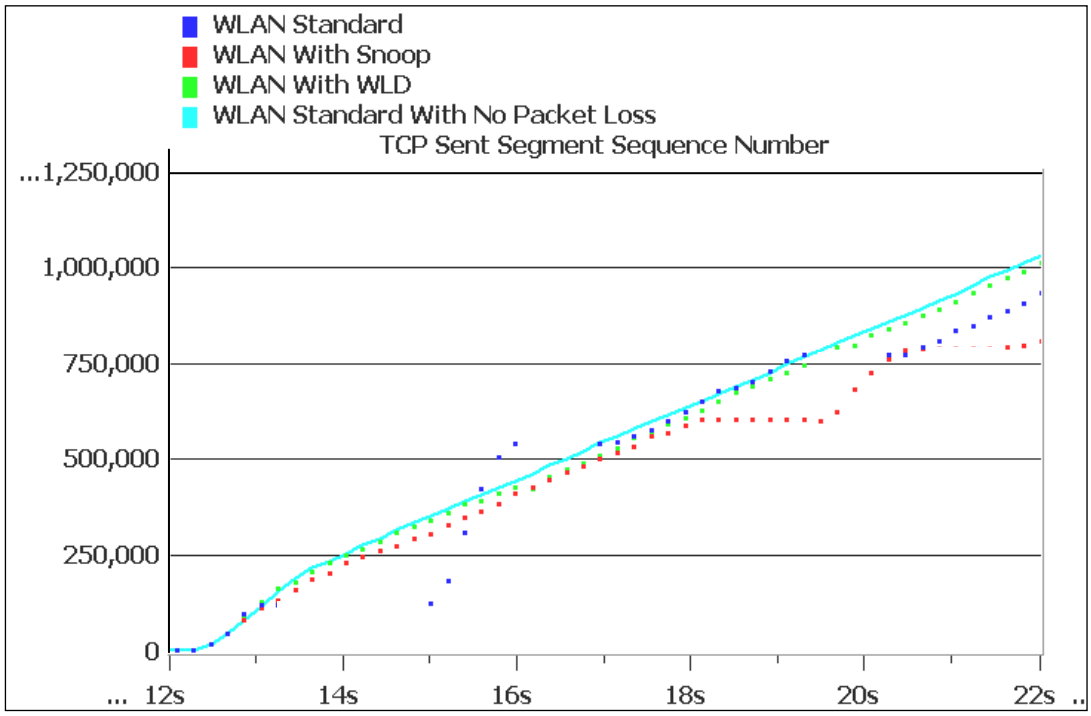

Figure 4-18 Compariosn TCP sent segmet sequence number responses for MH-5

Figure 4.19 shows the average number of cached packets, which reflect the amount of data transmitted within an RTT period of each TCP connection. The higher the packets in the Cached Table implies larger the TCP throughput for a given number of TCP connections. Since the WLD proxy only caches the TCP header information, it does not create more overhead at the base station than Snoop does.

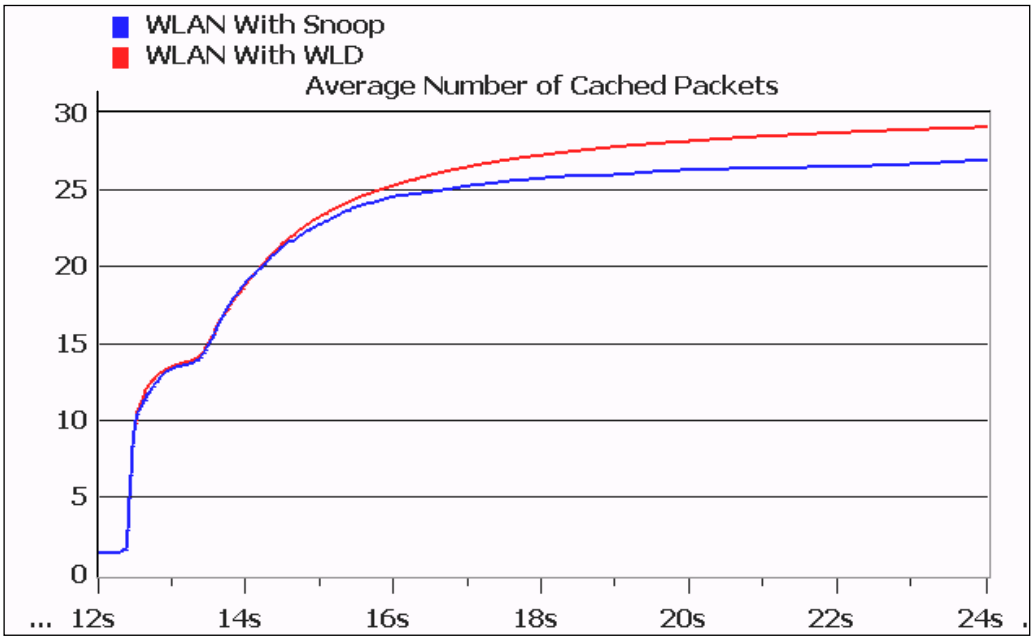

Figure 4-19 Average number of cached packets

Figure 4.20 shows the number of WLAN data traffic sent. It can be seen that WLAN with Snoop sent higher number of packets than the WLAN with WLD while achieving lower TCP throughput than the WLAN with WLD does. This implies the Snoop wastes the valuable network resources by competing for retransmission with the TCP sender.

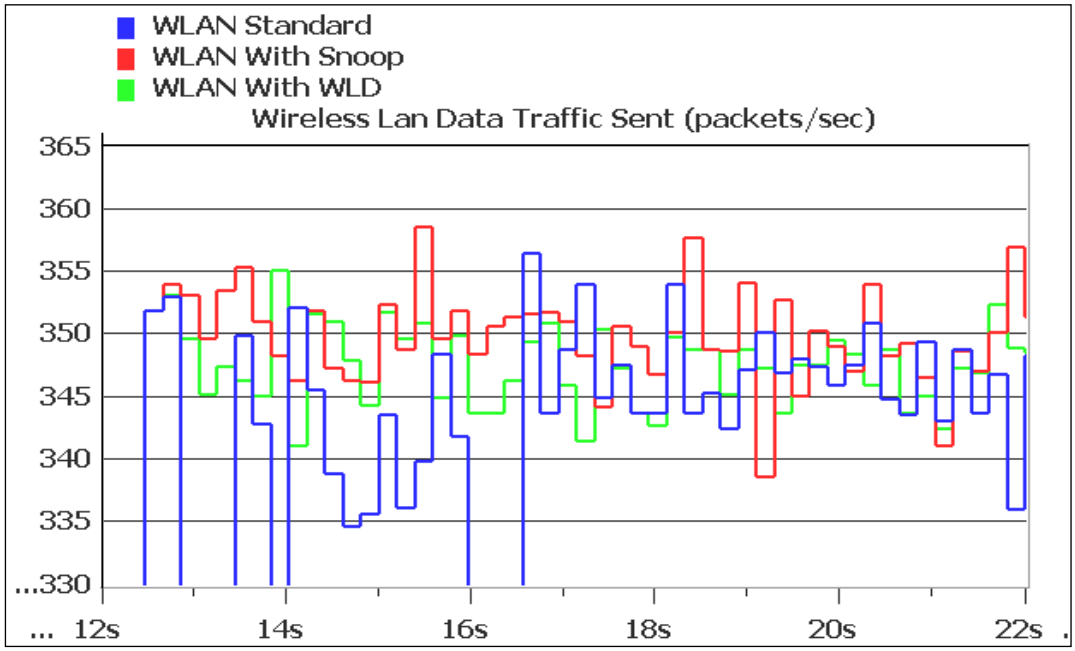

Figure 4-20 WLAN data traffic sent (packets/sec)

Our main objective is to detect all wireless packet losses and to make the TCP sender aware of this and perform wireless enhanced fast retransmit of those lost packets without reducing the CWND, taking the network utilization factor a value of 1.0. In order to see this effect, we have included the standard WLAN behavior without any packet losses in Figure 4.18, and it can be seen that the WLD proxy performs as expected; WLD scheme closely follows the performance of standard WLAN without any packet losses. Even though the proposed scheme has significantly improved the TCP performance in accordance with the intuition behind it, we further investigate its ability to deal with more rigorous wireless environments, for example with high packet drops rate, in the next experiment scenario.

4.4.2 EXPERIMENT WITH AN 802.11 WLAN ETHERNET ROUTER

The network model used for this study is shown in Figure 4.21. The Local Area Network (LAN) is extended using an 802.11 WLAN Ethernet router that forms a WLAN together with MHs. The model consists of two Servers, and some fixed hosts (FHs) and MHs. Servers and FHs are connected to the WLAN router through switches using 10baseT point-to-point link model. The Application and Profile Configuration nodes are configured to generate different applications such as HTTP, FTP Database and Email. FHs are configured to utilize some of these services in parallel with MHs in order to make the network analyzes be realistic and to show that the proposed scheme can be implemented in any WLAN devices.

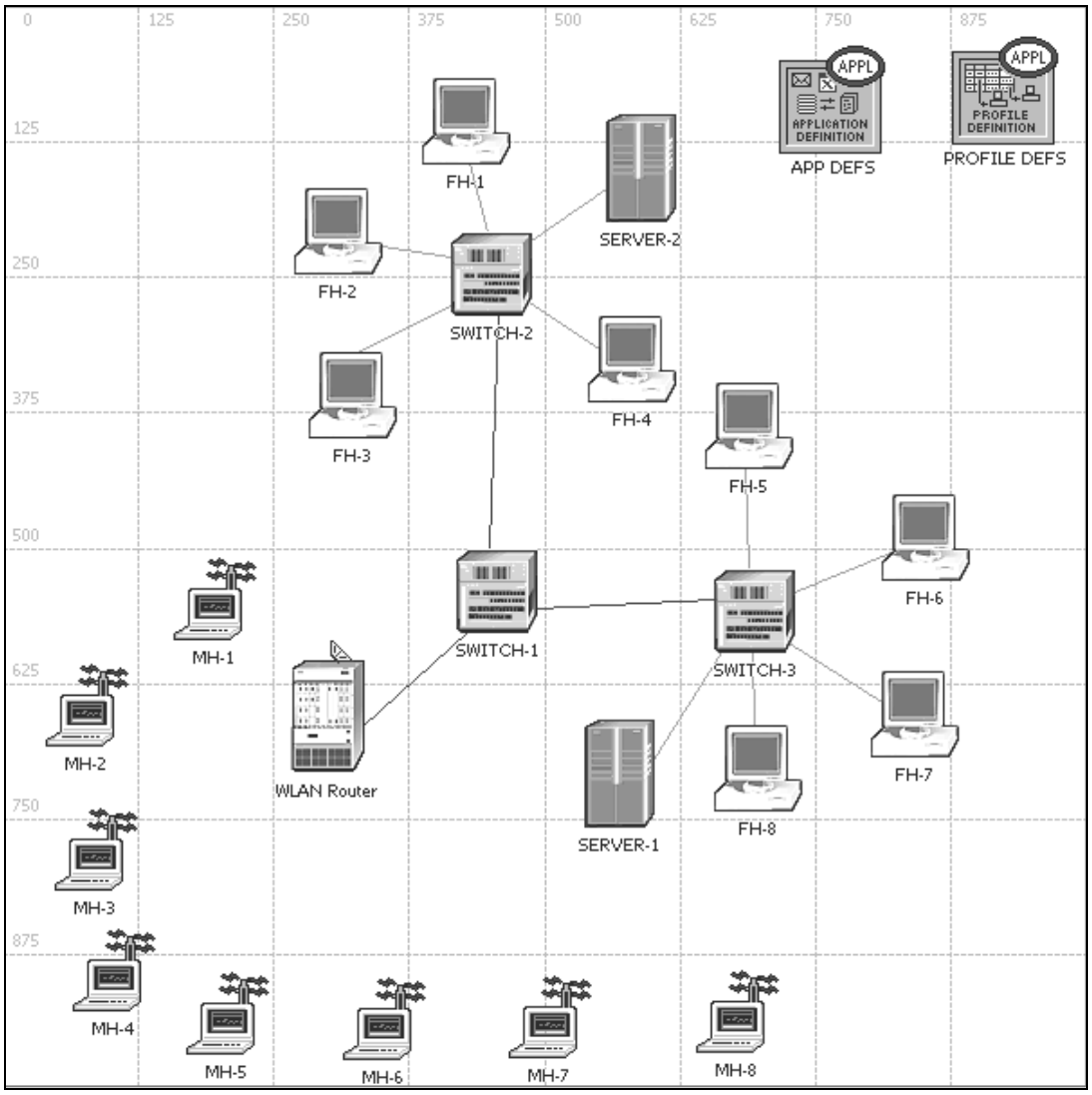

Figure 4-21 OPNET Network Model

All MHs are configured to download FTP files from Server-1 simultaneously. Packets coming from the Server-1 are dropped at the MAC layer of MH clients, using a uniform probability distribution and Table 4.4 shows the packet drop rates of MHs. The MAC layer is modified to generate bursty packet drops, as illustrated in Figure 4.22, so that the base station will perform its local retransmission and discard the packets once the threshold number of retry limit reaches. The packets that experience base station local retransmission before get successfully transmitted will imitate characteristics of wireless links, thereby impacting the end-to-end RTT. Figure 4.23 shows the OPNET representation of 802.11 WLAN Ethernet Router with the WLD proxy.

Mobile Host	1	2	3	4	5	6	7	8
Packet Drops Rate (%)	0	2	4	6	8	10	15	20

Table 4-4 Mobile Host Configuration

```
  rand_number = (int) op_dist_uniform (100.0);
  if(burst_error_length = -1)
  /*Initialize the burst error length to between 1 and 7, which is the short retry limit*/
     burst_error_length = (int)op_dist_uniform (7.0) +1 ;
  if (rand_number < recv_drop_rate AND (rcvd_frame_type = WlanC_Data))
 {
   /*We want to drop packets in burst. So wait until number of packets to be
     dropped equal to the burst error length*/
    if(busrt_counter < burst_error_length AND burst_error_occuring = 0)
     {
      busrt_counter++;
        if(busrt_counter = burst_error_length)
        burst_error_occuring = 1; /* this will allow burst packet drops */
     }
}
  if(burst_error_occuring = 1 AND (rcvd_frame_type == WlanC_Data))
{
    /*WE ONLY DROP DATA PACKETS*/
   busrt_counter--;
     accept = OPC_FALSE; /*This indicates data packet is corrupted. This packet will
                                 be discarded by the MAC layer processing*/
     if(busrt_counter = 0 )
     {
     burst_error_length = (int)op_dist_uniform (7.0) +1 ;
     burst_error_occuring = 0;
     }
}
```

Figure 4-22 Pseudo code for MAC layer packet drops

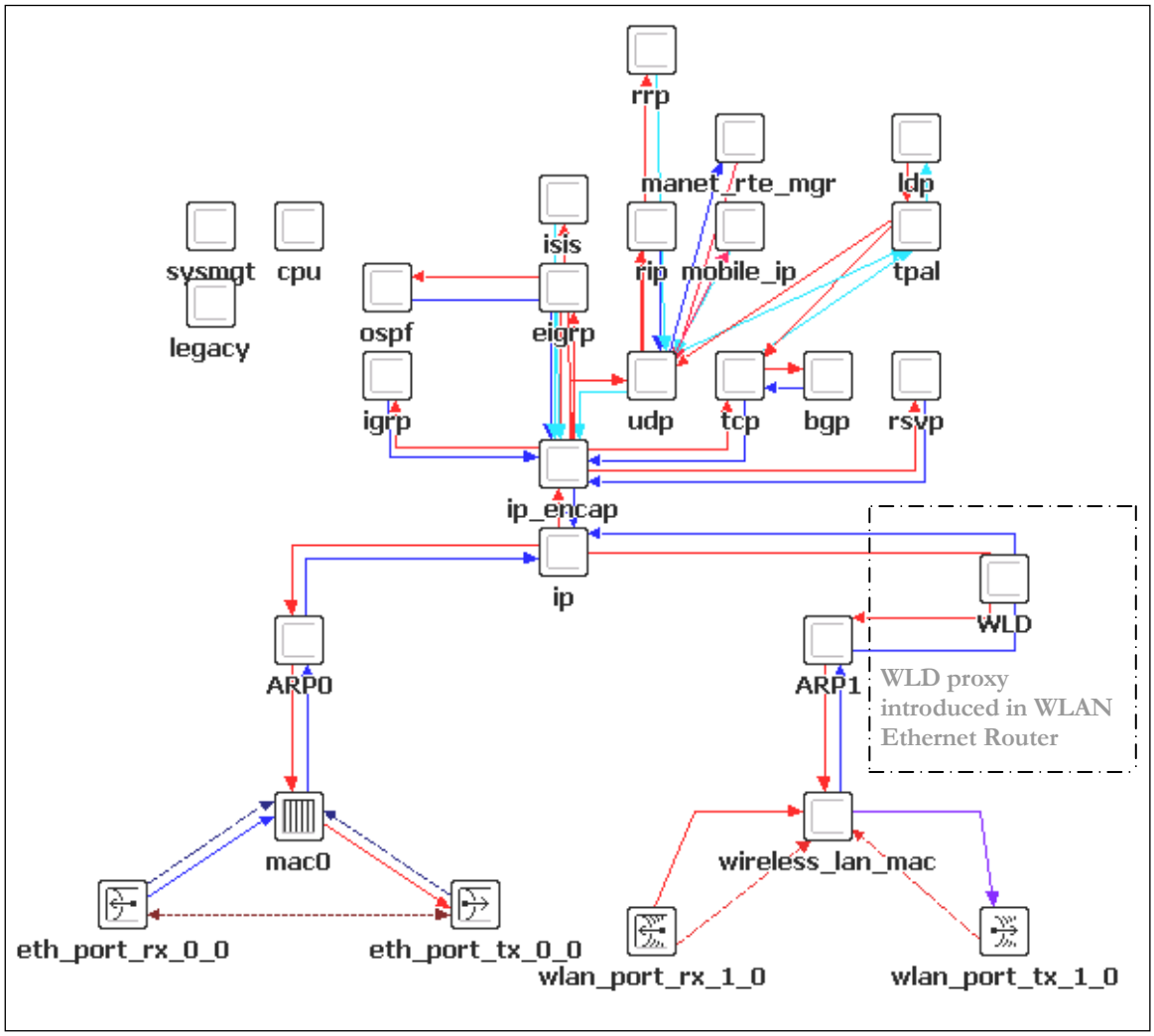

Figure 4-23 the OPNET representation of 802.11 WLAN Ethernet Router with WLD proxy

The WLAN Router is implemented, in turn, with a standard WLAN and a WLAN with WLD proxy to compare their relative performances. Modified TCP Reno with the default parameters is used in all simulation scenarios. Selected TCP Reno and WLAN parameter values are as given in Tables 4.2 and 4.3, respectively.

4.4.2.1 Simulation Results and Discussion

Figure 4.24 shows the TCP CWND size, sent segment sequence number, the MAC packet drops and the number of cached TCP headers responses during FTP file upload by MH-4 with the proposed scheme, WLAN with WLD, and the standard WLAN. Figures 4.25 and 4.26 show a snapshot of TCP CWND and TCP sent sequence number responses for MH-4. Comparison of

the TCP sent segment sequence number responses for MH-4, MH-5, MH-6, MH-7 and MH-8 is given in Figure 4.27.

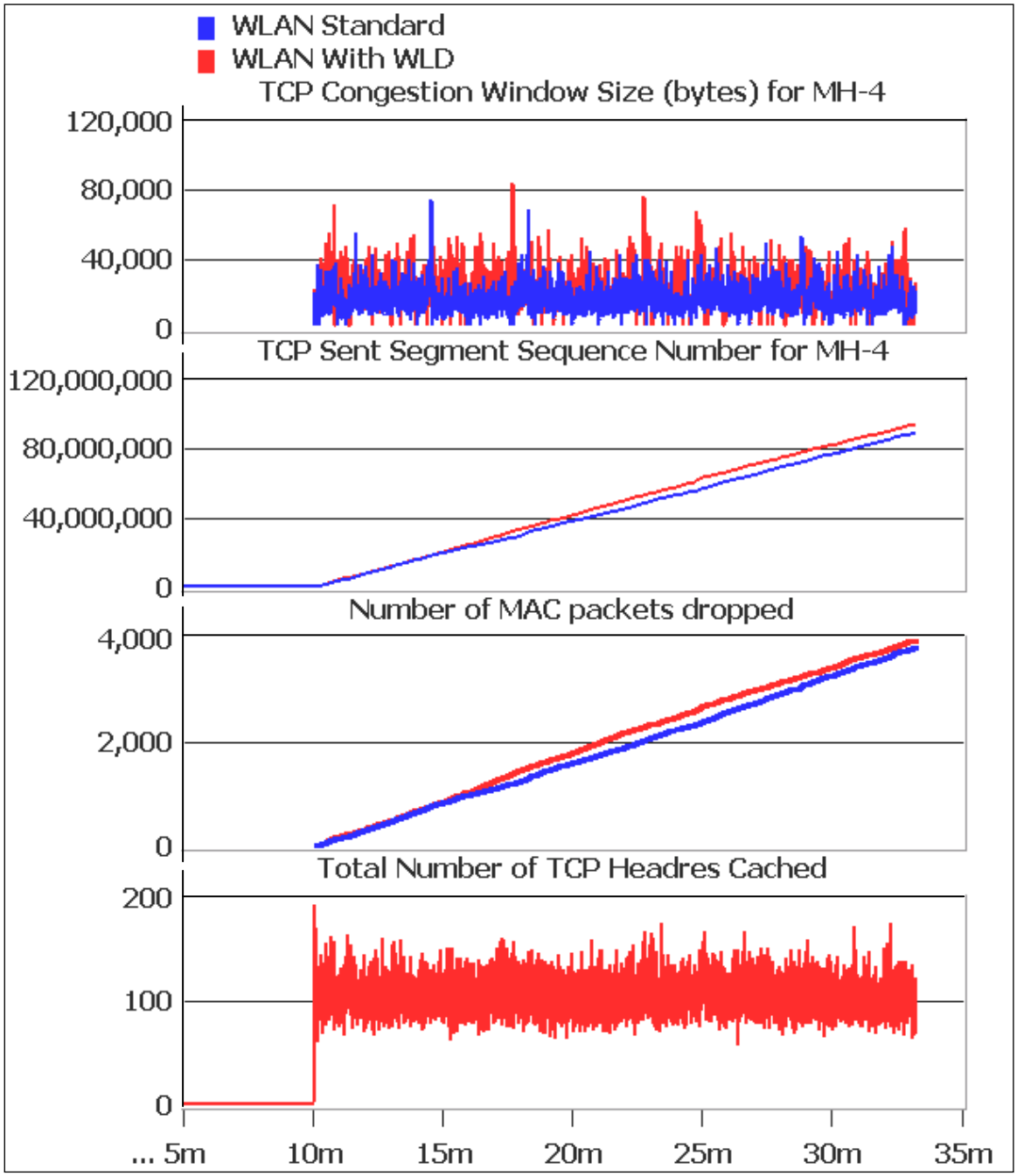

Figure 4-24 Responses during MH-4 client FTP file upload

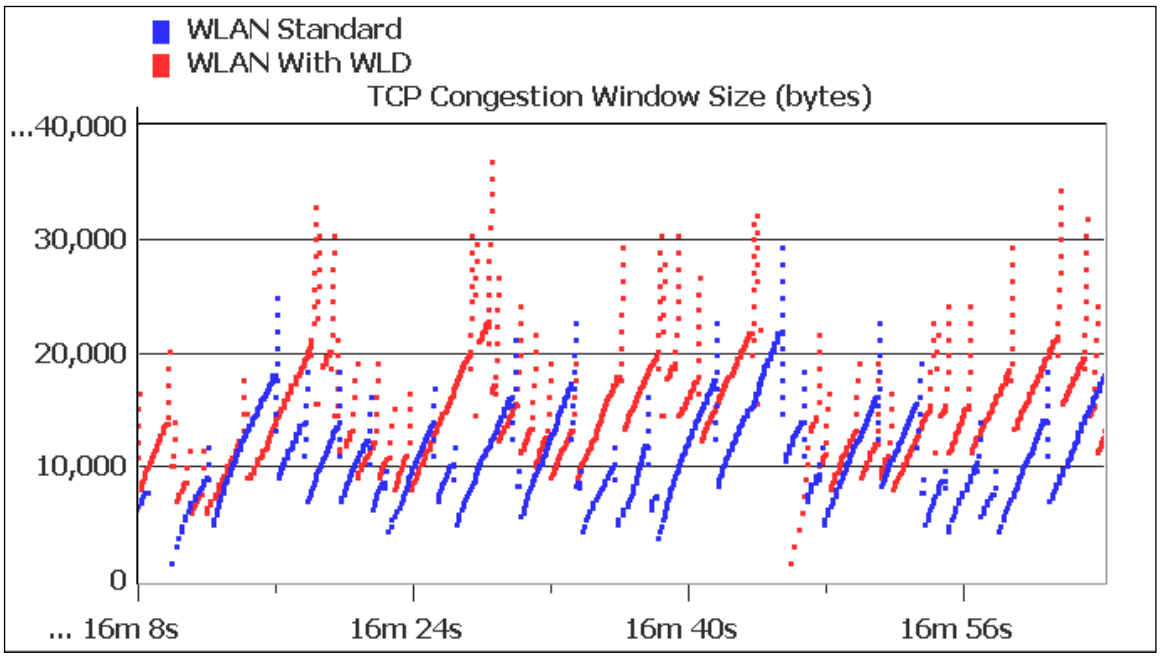

Figure 4-25 A Snapshot of TCP CWND response for MH-4

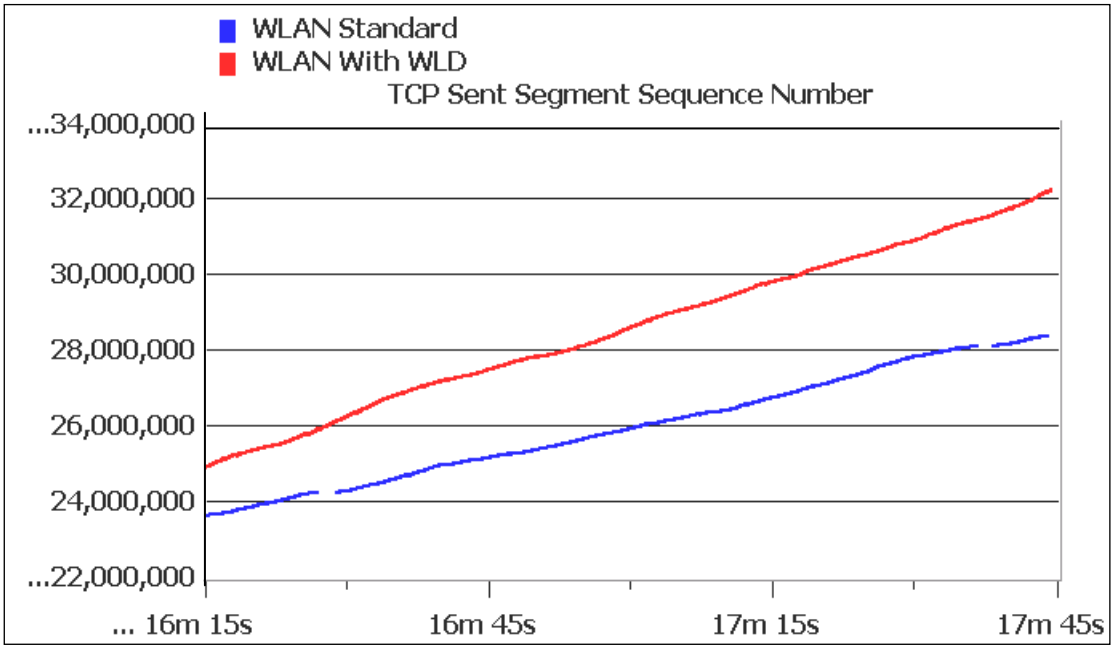

Figure 4-26 A Snapshot of TCP sent segment number response

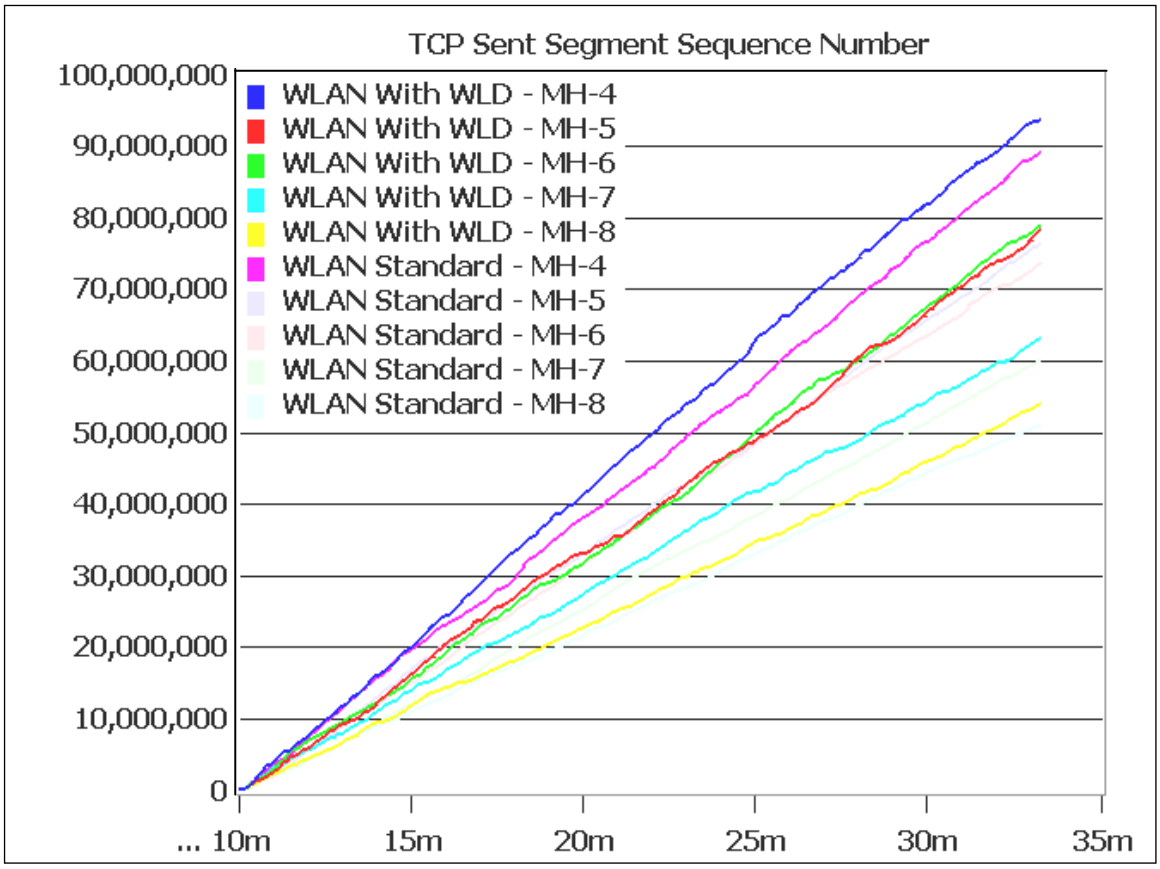

Figure 4-27 Comparison of TCP sent segment responses

From Figures 4.24 - 4.27, it can be seen that proposed scheme does not add much overhead to the WLAN Router and significantly improves the CWND and throughput responses compared to that of the standard TCP; there is a maximum of 353 TCP headers (7060 bytes) cached during the entire simulation period. The simulation was repeated many times with different seed values, which generated different MAC packet drop patterns, and the TCP mean throughput value was obtained with less than 5% error margin. Table 4.5 summarizes the TCP throughput performances. Figure 4.28 shows the TCP throughput improvement with the proposed scheme versus the MAC packet drop rates.

TCP Throughput (Kbps)	Mobile Host							
	1	2	3	4	5	6	7	8
Standard WLAN	1480	747	524	429	356	307	251	214
WLAN with WLD Proxy	1246	806	569	473	406	349	278	234

Table 4-5 Sumarry of TCP throughput performance

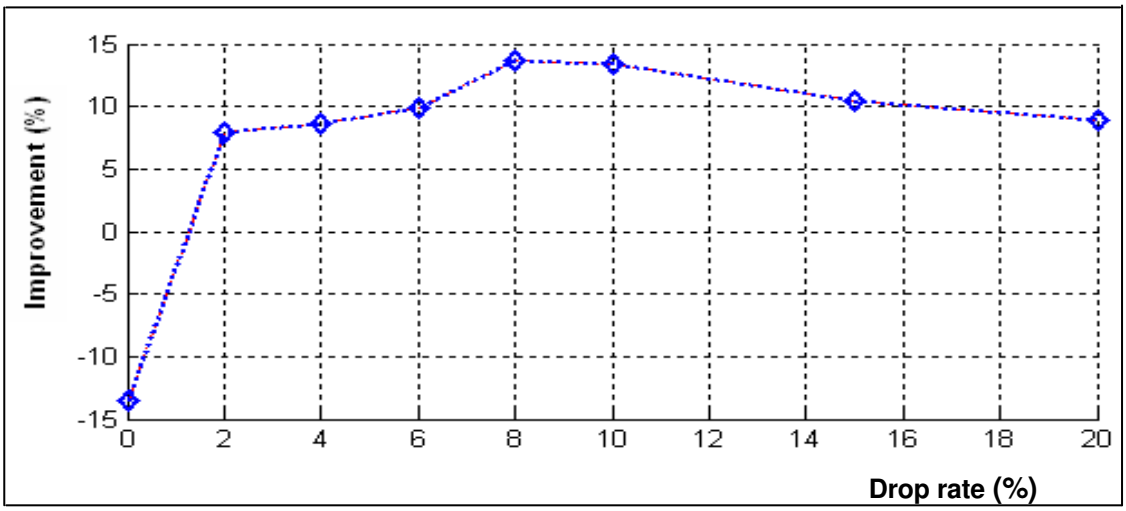

Figure 4-28 TCP Throughput improvement Vs Packet drop rates

From Table 4.5 and Figure 4.28, it can be observed that the proposed scheme has improved the TCP throughput performance of MHs with MAC packet drops. It should also be noticed that the proposed scheme has decreased the TCP throughput performance of MH-1, which does not drop any MAC packets. It demonstrates that TCP with enhanced fast retransmit and recovery algorithms to handle wireless packet drops is much fairer than the standard TCP Reno implementation. With the proposed scheme, the available BW across the wireless medium is more fairly shared among the 802.11 mobile hosts.

4.5 CONCLUSIONS AND FUTURE WORK

A new RNF scheme was presented that detects and distinguishes wireless packet losses from congestion related packet losses. TCP Reno was modified to distinguish wireless packet losses from the congestion losses, with the aid of the RNF flag, and to trigger the enhanced fast

retransmit and recovery algorithm accordingly. The standard TCP header was slightly modified to achieve this.

The RNF scheme was implemented in an 802.11 WLAN Server, with the bandwidth utilization factor $\alpha = 1.0$, in OPNET. The simulation results showed that it improved the TCP performance significantly compared to that of both the standard WLAN and the WLAN with the Snoop proxy. In particular, it detected all the wireless losses and enabled the TCP sender to trigger wireless enhanced fast retransmit and recovery algorithm to recover from those lost packets without reducing the congestion window.

It was also implemented in an 802.11 WLAN Ethernet Router in OPNET, with the bandwidth utilization factor $\alpha = 0.75$. Simulation results showed that the proposed scheme improved the TCP performance significantly compared to that of the standard WLAN. It also enabled the modified TCP Reno to distinguish wireless packet losses from the congestion-related losses and to trigger the wireless enhanced fast retransmit and fast recovery mechanisms to quickly recover from wireless packet losses. However, it could not completely avoid spurious TCP timeouts. The effects of spurious TCP timeouts can be minimized enabling the TCP sender to distinguish spurious timeouts from the normal timeouts. We leave this for our future work.

Simulation results also showed that the proposed scheme can handle multiple nodes and multiple TCP connections, and utilized the available network resources efficiently and fairly by adapting to the network characteristics. The advantage of this scheme is that it does not inject any additional packet into the network to provide feedback when it detects wireless packet losses; it does not compete for bandwidth and only sets the RNF flag of the ACK in case of a wireless detection. Finally, the proposed scheme only requires the software changes and can be easily implemented in real time network. Further validation of the scheme with different TCP flavors in different networks is left for future work.

Chapter V.

RNC Feedback Scheme for TCP Enhancement over a UMTS Network

Universal Mobile Telecommunications System (UMTS) is one of the most significant advances in the evolution of telecommunications into Third Generation (3G) networks. UMTS allows many more applications to be introduced to a worldwide base of users and provides a vital link between today's multiple Global System for Mobile Communications (GSM) and the ultimate single worldwide standard for all mobile telecommunications, International Telecommunications-2000 (IMT-2000). Development of advanced 3G networks and services makes it necessary to find a way of improving TCP's efficiency and resource utilization. TCP optimization for wireless networks to deal with packet losses due to fading, shadowing and contention should preferably maintain TCP end-to-end semantics with minimal dependence on intermediate nodes. Previous research on this issue suggests that TCP needs radio network feedback to distinguish wireless related losses from congestion related losses.

In this Chapter, we first give an overview of UMTS technology and, based on the RNF mechanism implemented in 802.11 WLAN environments, devise the radio network control (RNC) feedback mechanism in Sections 5.1 and 5.2, respectively. The RNC mechanism requires only a minimal change to the standard TCP implementation. Section 5.3 provides an incentive to the RNC development and outlines the advantages of the RNC mechanism over the RNF. The RNC mechanism is implemented in a UMTS network and the TCP performance with the RNC proxy is analyzed and compared with that of the standard UMTS in Section 5.4. The conclusions drawn and directions for future work are outlined in Section 5.5

5.1 AN OVERVIEW OF UMTS TECHNOLOGY

Third Generation (3G) technology is revolutionizing the capabilities of mobile communications. 3G networks are the next generation of mobile cellular networks, and their

origin is an initiative of the International Telecommunication Union (ITU). The main objective is to provide high-speed and high-BW wireless services to support a wide range of advanced applications, specially tailored for mobile personal communication such as telephony, paging, messaging and Internet access. 3G mobile networks are expected to provide more enhanced services than are possible over existing cellular systems, including higher bit rates services and greater capacity and service capability.

Universal Mobile Telecommunications System (UMTS) is one of the main 3G wireless technologies developed by the European Telecommunications Standards Institute (ETSI) within the IMT-2000 framework proposed by the ITU [Samukic, 1998]. UMTS evolved from global systems for mobile communications (GSM) and supports both existing services and offering new services including multimedia and access to the Internet with a speed of up to 2 Mbps; it provides network and service infrastructure convergence for fixed and mobile networks by allowing network operators to flexibly and efficiently offer services to customers irrespective of their access means [Mason, Cullen, & Lobley, 1996]. Currently, the Third Generation Partnership Project (3GPP) [3GPP], formed by a cooperation of standards organization, is in charge of developing UMTS technical specifications. UMTS systems have already been deployed in most European countries, although new and advance terminals, as well as many specifications, are still under development.

5.1.1 UMTS ARCHITECTURE

A UMTS network consists of three interacting domains: Core Network (CN), UMTS Terrestrial Radio Access Network (UTRAN) and User Equipment (UE). Figure 5.1 [Jamalipour, 2003] shows the UMTS architecture based on 3G TS 25.401 UTRAN Overall Description. The packet domain CN includes the serving GPRS support node (SGSN) and the gateway GPRS support node (GGSN). The GPRS support nodes (GSNs) include all GPRS functionality needed to support GSM and UMTS packet services. The SGSN monitors user location and performs security functions and access control. The GGSN contains routing information for packet-switched (PS) attached users and provides internetworking with external PS networks, such as packet data network (PDN).

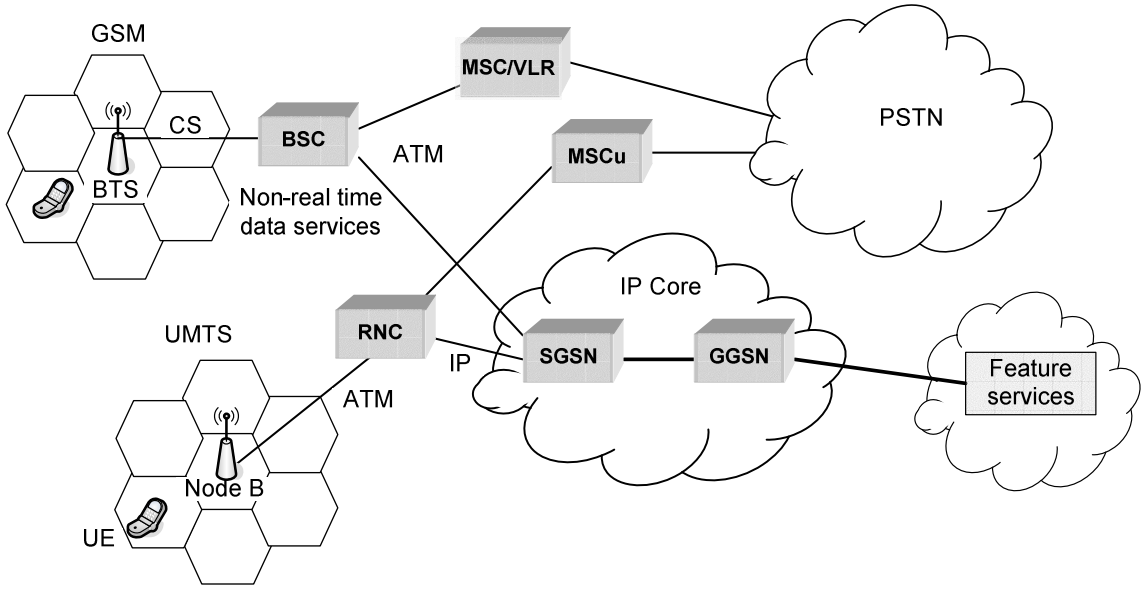

Figure 5-1 The UMTS network architecture

At the packet domain CN, UMTS mainly reuses the GSM/GPRS network elements. The MSC_U performs transcoding and bridges the cellular network to the public switched telephone network (PSTN). The circuit-switched (CS) core network includes the mobile switching center (MSC)/visitor location register (VLR). The MSC/VLR is used in the packet domain architecture to efficiently coordinate PS and CS services and functionality. Connection of the UMTS user terminal to the public telephony network is provided via an UMTS-type MSC, shown as MSC_U in the Figure 5.1. Access network domain, UTRAN manages specifications of the access technology of the UMTS, the wideband Code Division Multiple Access (W-CDMA). The BTS in the UMTS network is called a Node B and BSC uses the new name of radio network controller (RNC). The RNC provides data link layer services and the Node B supplies the physical (radio) channel access. User equipment (UE) is the end user UMTS cellular phone and provides data and voice services to the system users.

5.1.2 UTRAN ARCHITECTURE

Figure 5.3 provides an overview of UTRAN architecture showing the relationship between the CN and the UTRAN. Both CN and UTRAN are designed independently and are connected to each other through a set of standard interfaces. The UMTS core network consists of CS service domain and PS service domain, which are responsible for providing appropriate services to the CS traffic and PS traffic.

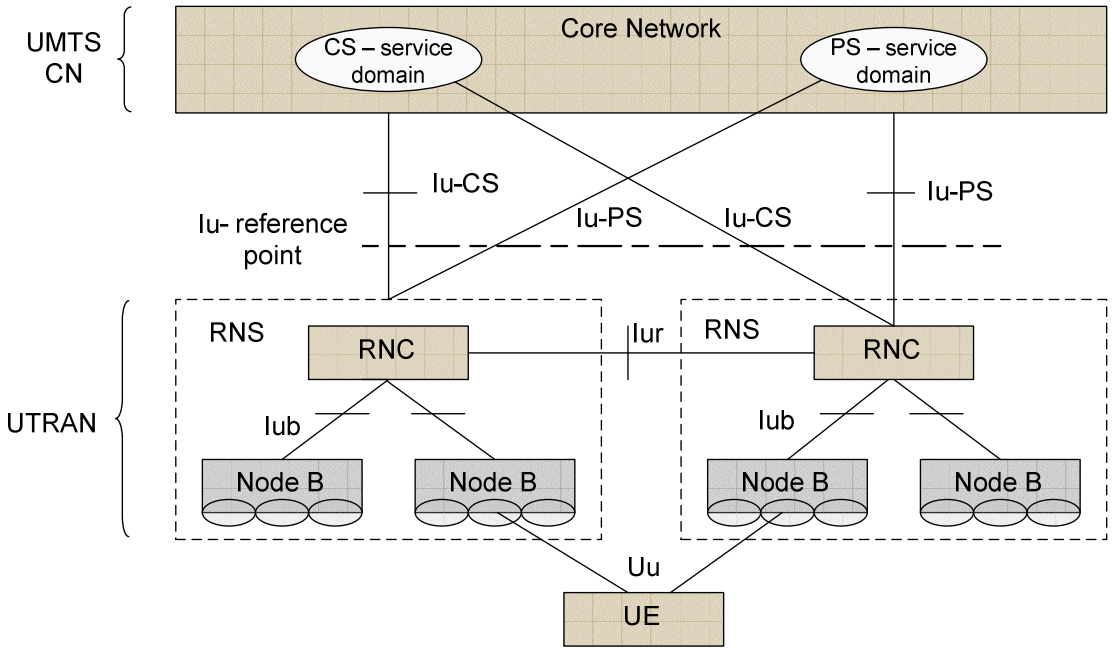

Figure 5-2 UTRAN architecture

The UTRAN consists of a set of Radio Network Subsystems (RNS) connected to CN through Iu interfaces. An RNS consists of a Radio Network Controller (RNC) and one or more Node-Bs. An RNC manages a set of base stations, referred as Node-Bs, through Iub interfaces. The RNC exerts admission control for new mobiles or services attempting to use the Node-B. Admission control ensures that mobiles are only allocated radio resources up to what the network has available. The RNC is also responsible for the Handover decisions that require signaling to the UE. Each UE has exactly one Serving RNC and can have one or more Drift RNCs, where the mobile physical layer communications terminate. Drift RNCs communicate with the Serving RNC via the Iur interface. A Drift RNC may also be the Serving RNC where no soft handover activity is in progress. Each Node B is responsible for connecting many end user terminals, UEs, to the UTRAN through the Uu interface. Detail description of UMTS access technology is given in the following Sections.

5.1.3 WIDEBAND CODE DIVISION MULTIPLE ACCESS (W-CDMA)

W-CDMA defines the air interface access of the UMTS network and is termed as UTRA. Unlike GSM and GPRS, which uses time division multiple access (TDMA) and frequency division multiple access (FDMA), W-CDMA allows all users to transmit at the same time and to share the same radio frequency (RF) carrier. In W-CDMA, instead of dividing users by frequency

or time as shown in Figure 5.3, they are divided into codes and specific data streams are assigned to particular users. Each mobile user is uniquely identified by a specialized code and frequency.

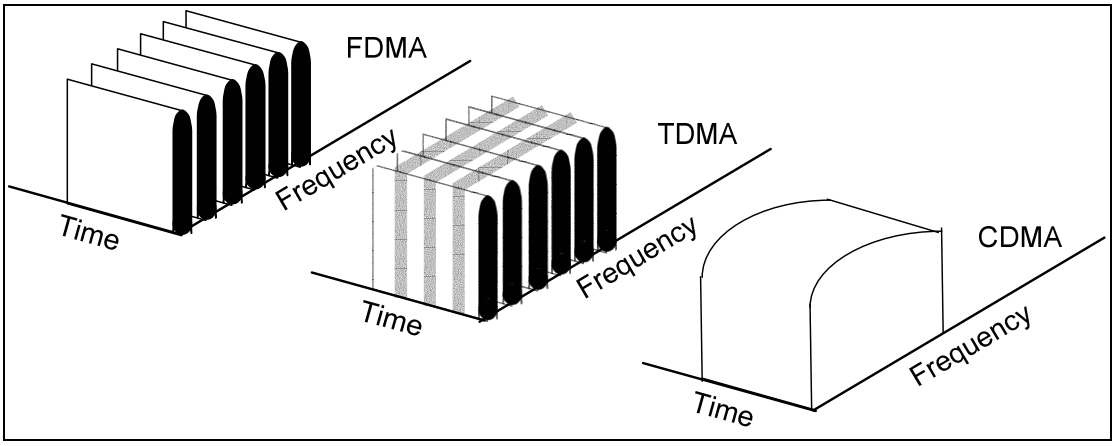

Figure 5-3 Comparison of multiple access schemes

In WCDMA Spread Spectrum technology, the information contents are spread by unique, digital codes (spreading sequence). The basic unit of a code sequence is one chip. Each user channel is uniquely identified by a code, which is a combination of a scrambling code and an orthogonal variable spreading factor (OVSF) code. Scrambling code is unique for each device and allows the recipient to identify from the other devices. The OVSF codes are used to separate traffic in a W-CDMA signal. W-CDMA uses a variable length code (4 to 512 chips), known as the spreading factor (SF). The SF may be updated as often as every 10 ms. This permits the overall data capacity of the system to be used more efficiently. Any user equipment that receives a transmitted data sequence and attempts to demodulate it using the wrong OSVF would interpret the information as noise. The noise, when integrated over time, will net to zero, which is an important property of the orthogonal codes in W-CDMA systems. The OVSF codes can be reused by each base station and user equipment within the same location, since the scrambling codes identify the transmitting device.

The CDMA digital mobile communication system has great potential power. In the CDMA system, system capacity is a soft capacity concept. For example, the system manager may raise the frame error rate to increase the available channels during peak hours of telephone traffic. Again, the CDMA system is a self-interference system, when its neighbor cells have less load, interference sent to the cell is smaller, so the capacity can be increased adequately.

One of the most important characteristics of W-CDMA is the fact that power is the common shared resource for users. In the downlink, the total transmitted power of an RF carrier is shared between the users transmitting from the base station by code division multiplexing (CDM). In the uplink, there is a maximum tolerable interference level at the base station receiver. This maximum interference power is shared between the transmitting user equipments in the cell, each contributing to the interference [Haardt et al., 2000]. Power as the common resource makes W-CDMA very flexible in handling mixed services and services with variable bit rate demands. Radio resource management is done by allocating power to each user to ensure that maximum interference is not exceeded. Reallocation of codes and time slots is normally not needed as the bit rate demand changes, i.e. the physical channel allocation remains unchanged even if the bit rate changes.

5.1.4 UMTS MODES OF OPERATION

ETSI special mobile group (SMG) defines two different mode of operation for the UTRA; frequency division duplex mode (FDD) [Dahlman, Gudmundson, Nilsson, & Skold, 1998], where the uplink and downlink are transmitted on different frequencies, and time division duplex (TDD) [Haardt et al., 2000], where the uplink and downlink are transmitted on the same frequency and are multiplexed in time. These modes are illustrated in Figure 5.4.

UTRA FDD is based on 5 MHz W-CDMA with a basic chip rate of 4.096 Mchips/s, corresponding to a bandwidth of approximately 5 MHz. Higher chips rates of 8.192 and 16.384 Mchips/s are also specified intended for future evolution of the W-CDMA air interface towards data rates higher than 2 Mbps [Dahlman, Gudmundson, Nilsson, & Skold, 1998]. The basic radio frame has a length of 10 ms, allowing for low delay speech and fast control messages, and is divided into 15 slots. W-CDMA carriers are located on a 200 kHz carrier grid with typical carrier spacing in the range of 4.2 – 5.0 MHz. Spreading factors vary from 256 to 4 for an FDD uplink and from 512 to 4 for an FDD downlink. With these spreading factors, data rates of up to 2 Mbps are attainable.

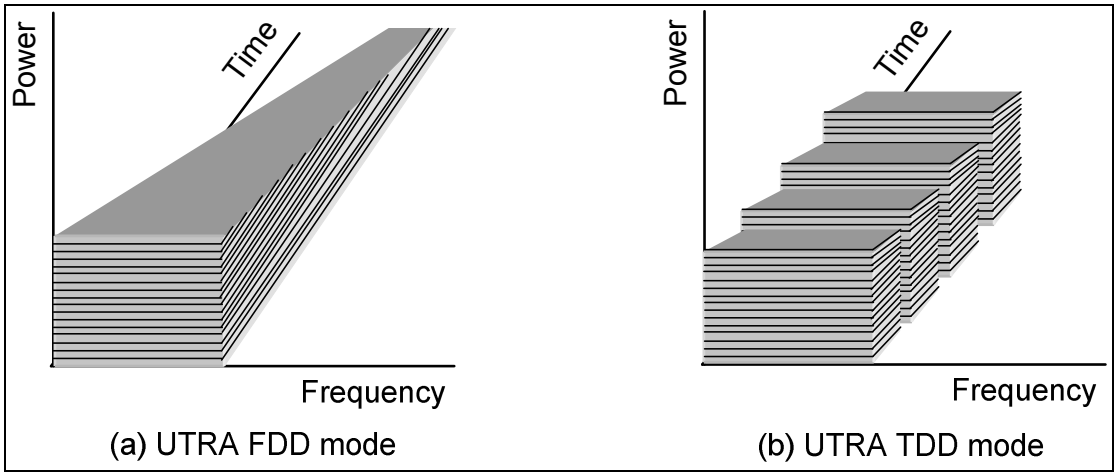

Figure 5-4 UTRA FDD and TDD modes of operation

UTRA TDD is based on TD/CDMA technology. During the subsequent ETSI process, the parameters of the TDD mode have been completely harmonized to the FDD mode. TDD is the only method that can flexibly share the system capacity to uplinks and downlinks [Jamalipour, 2003]. The main difference between the FDD and TDD modes is that the TDD mode includes an additional TDMA component, allowing for interference avoidance by means of dynamic channel allocations. Since in TDD, the uplink and downlink are transmitted on the same frequency it is possible to allocate different ratio of TDD time slots to uplinks and downlinks in accordance to the service requirements at a particular time. This flexibility is not available in an FDD mode, as a fixed amount of total system capacity is devoted to the uplink and the rest to the downlink.

5.1.5 UMTS RADIO INTERFACE PROTOCOL ARCHITECTURE

The UMTS radio interface protocol architecture is shown in Figure 5.5. The architecture consists of a control plane (C-plane) for signaling and user plane (U-plane) for data information transportation.

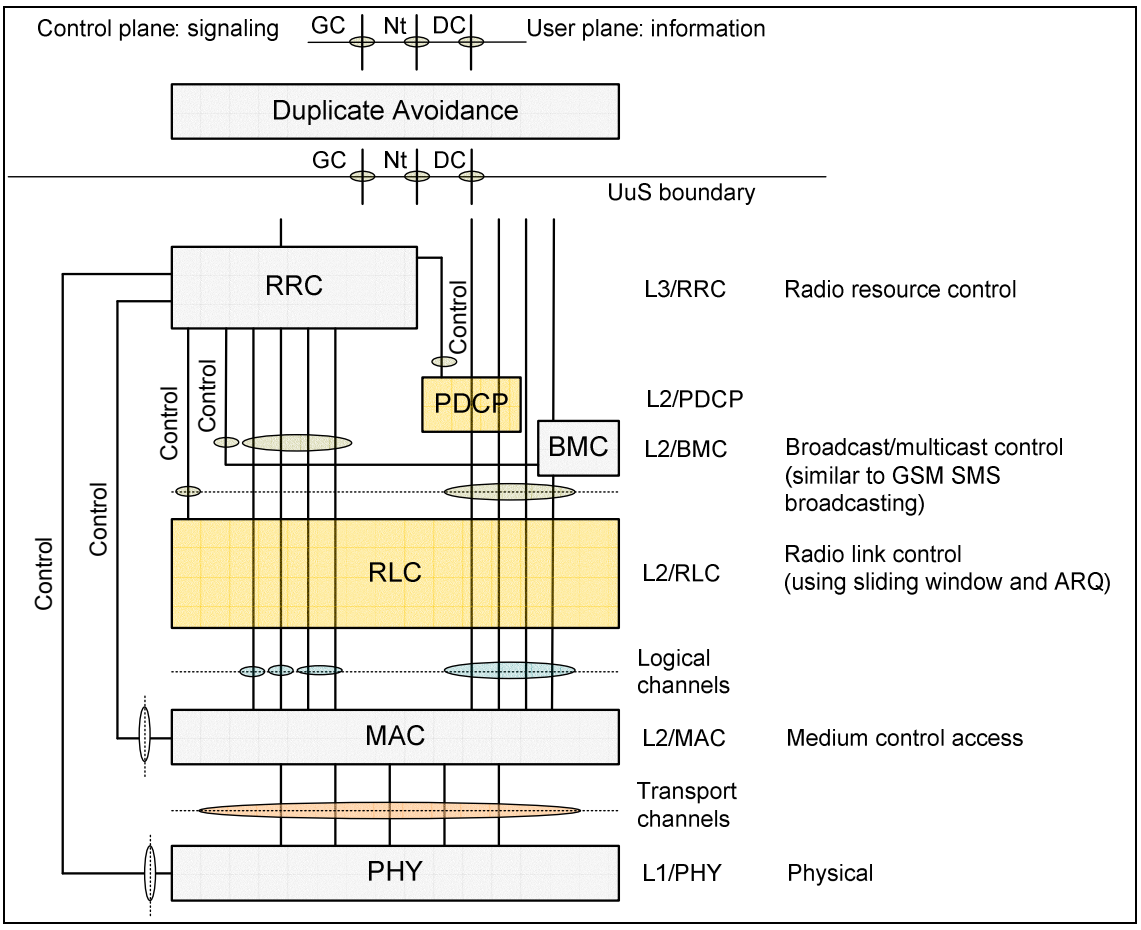

Figure 5-5 UMTS radio interface protocol architecture

The design of the radio interface protocol stack has focused on a clear structuring of the layers. The main functionality layers are physical layer (PHY), medium access control layer (MAC), radio link control layer (RLC), broadcast/multicast control layer (BMC), packet data convergence protocol layer (PDCP) and radio resource control layer (RRC). The services provided by each layer and the physical and logical channels at the physical layer and MAC layer are briefed, respectively, in the following.

The physical layer is responsible for the transmission of transport blocks over the air interface. This includes forward error correction (FEC), multiplexing of different transport channels on the same physical resources, rate matching, modulation, spreading and RF processing. The error indication in the physical layer is important for the realization of incremental redundancy protocols. Physical channels could be of one of the following types:

- *Random access channel* (RACH) is a contention based uplink transport channel for initial channel access to the network as well as for short data bursts. The RACH is always received from the entire cell and is characterized by a collision risk and by being transmitted using open loop power control.

- *Common packet channel* (CPCH) is a contention based used for transmission of bursty data traffic. This channel only exists in FDD mode and only in the uplink direction. The CPCH employs fast power control and is shared by the UE in a cell and therefore, it is a common resource.

- *Forward access channel* (FACH)

- *Downlink shared channel* (DSCH)

- *Broadcast channel (BCH)*

- *Paging channel* (PCH)

- *Dedicated channel* (DCH)

MAC layer maps the logical channels of the RLC on the transport channels provided by the physical layer. Its main functionality is multiplexing different data streams. The MAC layer is informed about resource allocations by the RRC and performs priority handling between different data flows that are mapped on the same physical resource. The logical channels provided by the MAC layer are as follows:

- Control channel (CCH)
  - Broadcast control channel (BCCH)
  - Paging control channel (PCCH)
  - Dedicated control channel (DCCH)
  - Common control channel (CCCH)

- Shared control channel (SHCCH)

- Traffic channel (TCH)
  - Dedicated traffic channel (DTCH)
  - ODMA dedicated traffic channel (ODTCH)
  - Common traffic channel (CTCH)

RLC layer provides transparent, unacknowledged, or acknowledged mode data transfer to the upper layers. The acknowledged mode transfer uses a sliding window protocol with selective reject-automatic repeat request (ARQ). The RLC also provides segmentation and retransmission services for both users and control data. It offers services to higher layers via service access points (SAP), which describes how the RLC layer handles the data packets. On the C-plane, the RLC services are used by the RRC layer for signaling transport. On the U-plane, the RLC services are used either by the service specific protocol layers (PDCP or BMC) or by the higher layer U-plane functions.

Packet data convergence protocol (PDCP) layer is located in the U-plane and provides header compression functions for network protocols such as TCP/IP and UDP/IP [RFC 2507] . It also handles transmission and reception of protocol data units (PDUs) using services provided by the RLC protocol and supports for SRNs loss-less relocation.

RRC layer handles the C-plane signaling of layer 3 between UTRAN and the UE. It is also responsible for controlling the available radio resources. This includes the assignment, reconfiguration, and release of radio resources as well as continuous control of the requested quality of service. The TDD mode requires additional features in the RRC layer such as dynamic channel allocation, handling of the outer loop power control, ad timing advance control.

5.1.6 NODE B

The Node B is the function within the UMTS network that provides the physical radio link between the user equipment and the network. UMTS uses Wideband Code Division Multiple Access (WCDMA) to carry the radio transmissions. A Node-B can support FDD, TDD or dual-mode operation. Node B connects with the UE via the W-CDMA Uu radio interface and with

the RNC via the Iub ATM based interface, and performs the conversion of data to and from the Uu radio interface. It determines the frame error rate (FER), based on the quality and strength of the connection, and transmits these data to the RNC as a measurement report for handover and macro diversity combining. The macro diversity combining is carried out independently, eliminating the need for additional transmission capacity in the Iub interface.

Node B also participates in power control as it enables the user equipment to adjust its power using downlink transmission power control (TPC) commands via an inner-loop power control on the basis of uplink TPC information. The UMTS standard specifies a downlink power adjustment procedure for adjusting the Node B transmitted power of the radio links in the active set. However, the standard leaves opens the specific method used to compute and apply the adjustment corrections.

5.1.7 RADIO NETWORK CONTROLLER (RNC)

The RNC handles protocol exchanges between Iu, Iur and Iub interfaces and is responsible for centralized operation and maintenance of the entire RNS with access to the operating subsystem. Since the interfaces are ATM based, the RNC switches ATM cells between them. The user's circuit switched and packet switched data coming from Iu-CS and Iu-PS interfaces are multiplexed together for multimedia transmission via Iur, Iub and Uu interfaces to and from the user equipment. The RNC uses the Iur interface to autonomously handle radio resource management (RRM), eliminating the burden from the core network. Serving control functions such as admission, RRC connection to the UE, congestion and handover/macro diversity are managed entirely by the serving RNC (SRNC). If another RNC is involved in the active connection through an Inter-RNC Soft Handover, it is declared a drift RNC and is responsible for the allocation of code resources.

5.1.8 3GGP RELEASE 5

The initial standards for UMTS were completed by 3GPP in April of 1999 and termed Release 1999 (R'99). These standards are the basis for a majority of the current commercially deployed UMTS systems previously discussed. In April of 2001, a follow up release to R'99 was standardized in 3GPP, termed Release 4 (Rel'4), which provided minor improvements of the UMTS transport, radio interface and architecture. In March 2002, Release 5 (Rel'5) [3GPP] of UMTS was completed which defined features such as the High Speed Data Packet Access (HSDPA) channel, the IP Multimedia Subsystem (IMS) and IP UTRAN that provide significant

spectral/network efficiency, performance and functionality advantages over the R'99 and Rel'4 standards.

The Rel'5 UMTS standards were developed such that the Rel'5 enhancements can co-exist on the same RF carrier as currently deployed R'99 UMTS. Thus, a current R'99 UMTS carrier can be upgraded to support legacy R'99 as well as new Rel'5 terminals in the same 5 MHz band. HSDPA is one of the key Rel'5 features that offers significantly higher data capacity and data user speeds on the downlink (theoretically up to 14 Mbps peak) compared to R'99 UMTS through the use of very dynamic adaptive modulation, coding and scheduling with Hybrid Automatic Retransmission Request (H-ARQ) processing. Through HSDPA, operators will benefit from a technology that will provide improved end-user experience for web access, file download and streaming services. Wireless Broadband access to the Internet, intranet and corporate LAN will benefit greatly from HSDPA. In addition to HSDPA, UMTS Rel'5 introduces the IP Multimedia System (IMS) architecture that promises to greatly enhance the end-user experience for integrated multimedia applications and offer the mobile operator an efficient means for offering such services. The IMS enables new and more advanced multimedia applications for operators (including VoIP), the ability for these services to interact and the ability to fully integrate real-time, near real-time as well as non-realtime services. UMTS Rel'5 also introduces the IP UTRAN concept to realize network efficiencies and reduce network costs.

5.2 RADIO NETWORK CONTROLLER FEEDBACK MECHANISM

A study of Wireless TCP proposals with proxy servers in the GPRS network [Rendon, Casadevall, & Carrasco, 2002] analyses the Split mechanism [Bakre & Badrinath, 1995] and the Snoop protocol [Balakrishnan, Padmanabhan, Seshan, & Katz, 1997] and concludes that the Snoop protocol does not perform well because of the high delays in the GPRS radio channel and the Split mechanism slightly improves the TCP throughput. Snoop is a link-layer protocol that retransmits the lost packets locally, trying to hide packet losses across the wireless link from the TCP sender, while the Split mechanism attempts to separate loss recovery over the wireless link from that over the wire-line network. A cross-layer congestion avoidance scheme in [Kliazovich & Granelli, 2005] gathers network capacity information such as BW and delay at the link layer and, based on these measurements, it adjusts the outgoing data stream. Link layer proposals [Wong & Leung, 1999] try to hide packet losses across the wireless link from the TCP sender by employing error correction using techniques such as FEC and retransmitting the lost packets

locally in response to the ARQ message. Adaptive-TCP [Seok, Youm, kim, & Kang, 2003] is a TCP-aware link layer protocol and performs local retransmission, sender freezing and flow control in order to hide the wireless environment from the TCP sender.

The abovementioned protocols attempt to either hide or separate the wireless effects from that of traditional wire-line network. However, they all fail to completely recover from the wireless effects [Balakrishnan, Padmanabhan, Seshan, & Katz, 1997; Sinha, Nandagopal, Venkitaraman, Sivakumar, & Bharghavan, 1999]. In this Chapter, we propose a new technique that utilizes the RNC feedback (RNC-FB) to completely distinguish the wireless packet losses form the congestion related losses.

5.2.1 PROPOSED RNC FEEDBACK MECHANISM

Previous studies show that minimizing the wireless environment effects could improve the TCP performance. In reality, they do optimize the link layer protocols and this in effect improves the TCP performance. The actual TCP improvements should be achieved by tuning the TCP itself to utilize the available network resources efficiently in both wire-line and wireless environments. We propose a scheme that monitors the UMTS network radio interface and notifies the TCP sender of any effects caused by the wireless link. The TCP End-to-End semantic is maintained but it is modified in order to adapt to the characteristics of the wireless environment.

An IP packet entering the UMTS network is double encapsulated, as shown in Figure 5.6, to cross the UMTS network. The GPRS Tunneling Protocol (GTP) sets up the GTP tunnels between the GGSN and SGSN. When this packet reaches the RNC node, the original IP datagram is obtained at the GTP layer of the RNC protocol stack, shown in Figure 5.7, and passed on to the RNC layer for delivery to the destination UE. Our proposed scheme modifies the GTP layer of RNC protocol stack in order to be able to monitor the IP datagram flows and to extract TCP header information, assuming that the IP datagram is not encrypted. Extracted TCP header information is maintained in a cache table and used to observe wireless environment effects and prepare the RNC feedback to the TCP sender. The functionality of the GTP layer and the required modifications is briefly explained next.

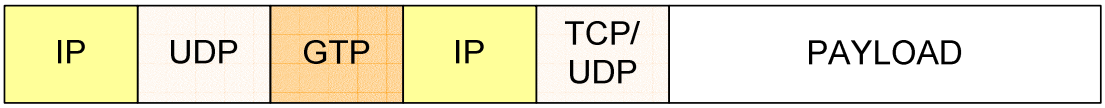

Figure 5-6 IP Datagram double encapsulation

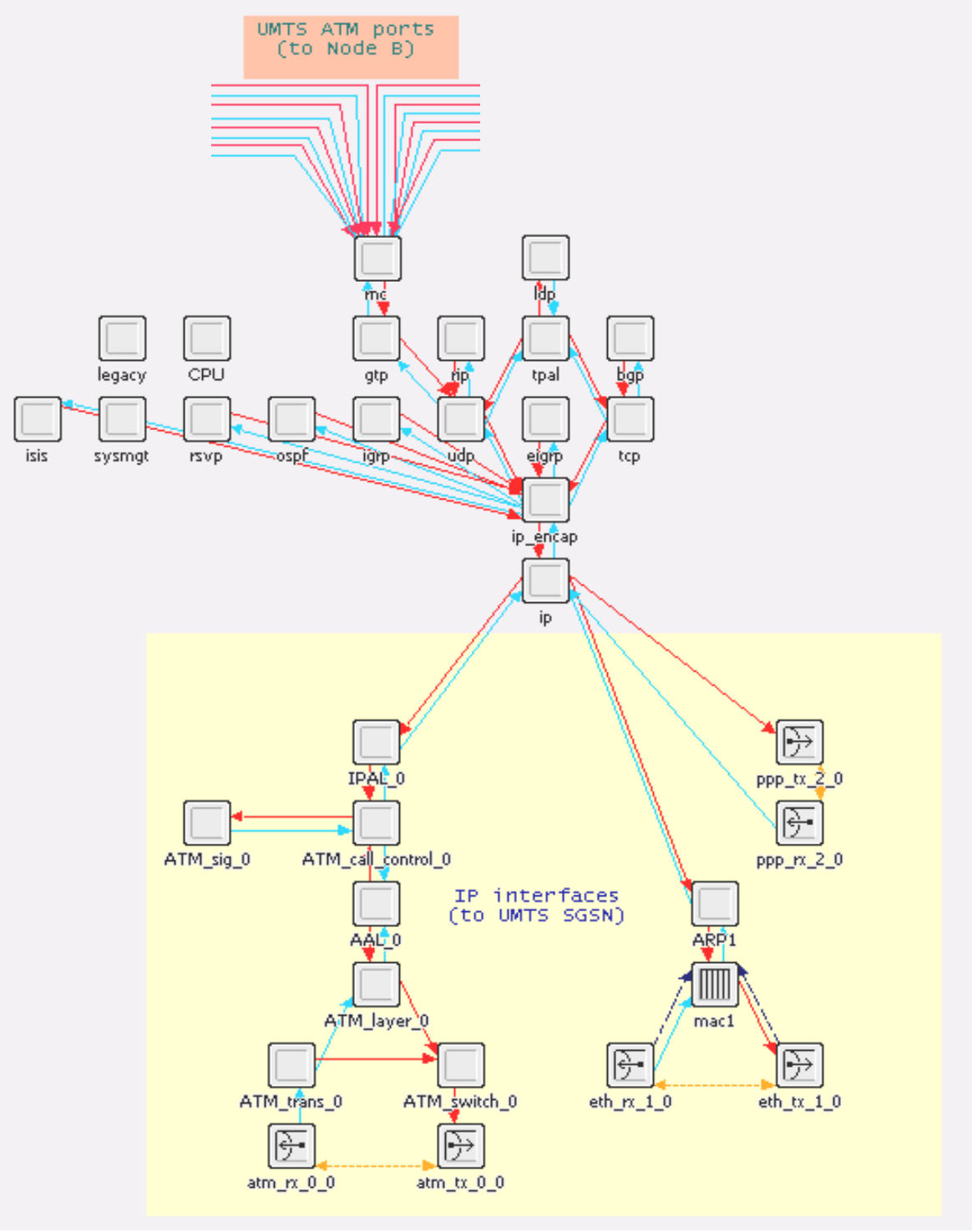

Figure 5-7 OPNET representation of RNC node model

When receiving a packet from the lower layer (UDP Layer), the GTP layer in the RNC protocol stack takes one of the two possible actions, depending on the packet types:

- If the packet is a GTP signaling message, GTP processes it locally and acts accordingly, otherwise decapsulates and forwards the packet to the higher layer.

- If a packet is delivered by the upper layer (RNC layer), it encapsulates and tunnels the packet or directly delivers the packet to the UDP layer if the Iur interface implementation is ready.

We modify the implementations of both GTP encapsulation and decapsulation states in the GTP process model shown in Figure 5.8 to generate RNC feedback.

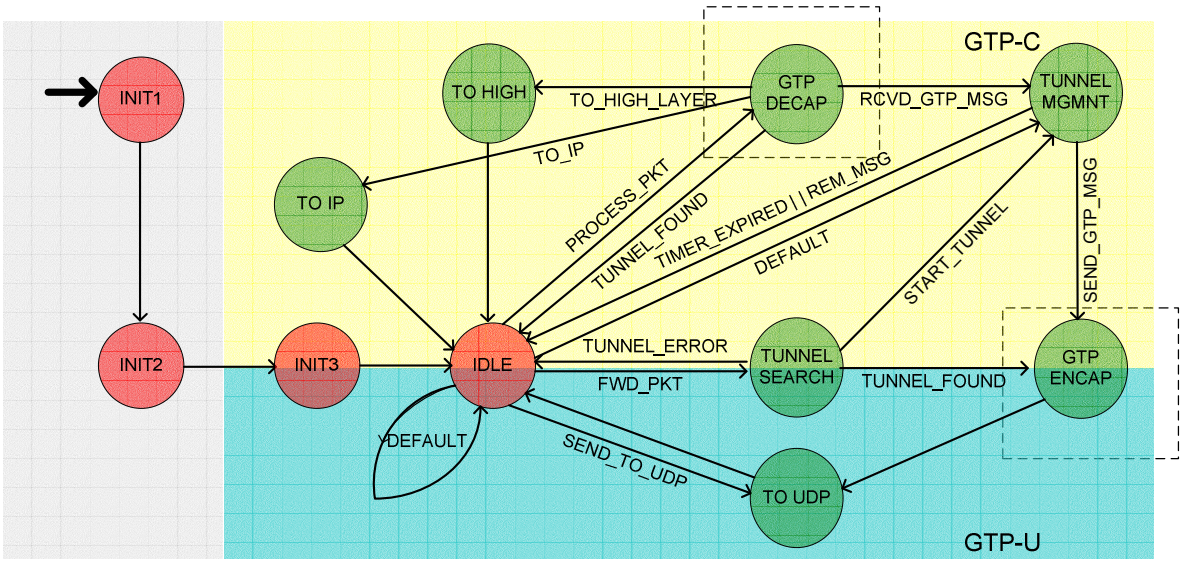

Figure 5-8 GTP process model

GTP encapsulation state is modified to monitor all the packet flows and to maintain a Cache Table with the TCP header information, including the TCP connection information to support multiple TCP connections. GTP decapsulation state is modified to utilize the cached TCP header information to provide RNC feedback to the TCP sender as follows; it monitors all the packet flows and uses the Cache Table to detect wireless packet losses. If a wireless packet loss is detected, it notifies the TCP sender of this by utilizing the control bit next to the CWR flag in the reserved field of the TCP header, called the Radio Network Feedback (RNF) flag. The standard TCP should be tuned to accommodate this effect so as to avoid unnecessary congestion window reduction and spurious timeouts due to wireless environment effects.

Impact of transmission errors on TCP reduces the network resource utilization; it forces the TCP to reduce the CWND by invoking unnecessary congestion control measures, resulting in poor throughput. TCP is modified to accept the RNF control flag and is tuned to perform wireless fast retransmit without reducing the congestion window if the RNF flag is set. Wireless fast retransmit is triggered when the TCP sender receives two duplicate ACKs with RNF flag set to quickly recover from the wireless packet losses. The sooner the fast retransmit occurs, the better TCP performs because the TCP recovery phase ends and the congestion avoidance phase is entered sooner. It also minimizes the spurious TCP timeouts, resulting in further TCP performance improvement. It should be noted that this scheme does not introduce a proxy at the RNC node. It requires only the software change at the RNC node and does not add much overhead to the RNC node since only the TCP header information is cached.

5.3 OPNET IMPLEMENTATION OF THE RNC FEEDBACK MECHANISM

There are eleven states in OPNET implementation of GTP process model shown in Figure 5.8. Modifications of "GTP DECAP" and "GTP ENCAP" states are required to implement the proposed RNC feedback (RNC-FB) scheme. Figures 5.9 and 5.10 show the flow controls for modifying the "GTP DECAP" and "GTP ENCAP" states respectively. Note that our modification to the "GTP DECAP" state looks only for the IP datagram flow to extract TCP header information because the information flow between SGSN and UE is not only data packets. There is also some UMTS management frames exchanged between these nodes. Actually, before any data flow can happen, first the UE has to complete its GMM GPRS attachment with the CN, and then go through packet data protocol (PDP) Context Activation with the SGSN node for the Quality of Service (QoS) class of the data flow.

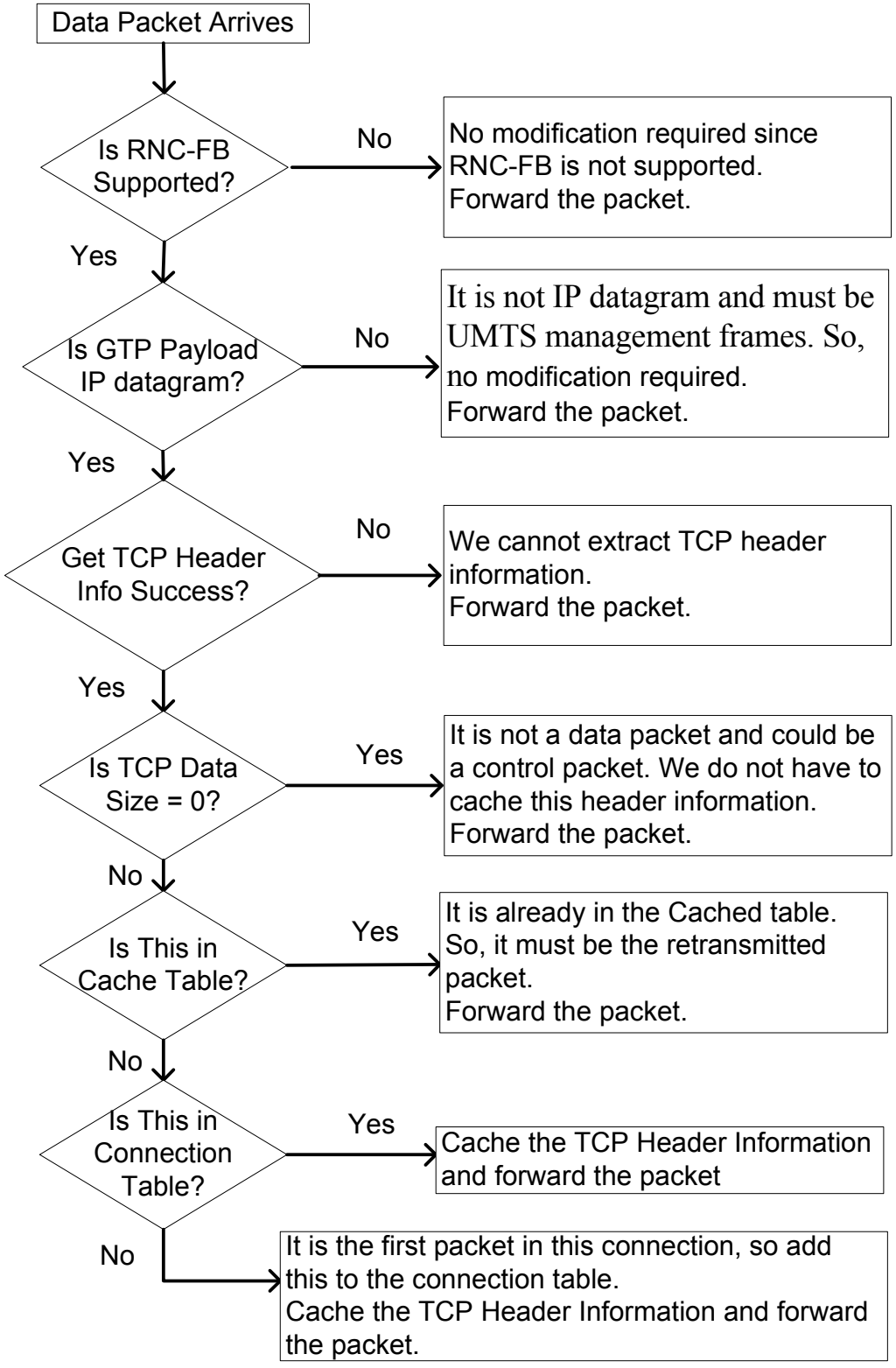

Figure 5-9 Modifications of "GTP DECAP" state

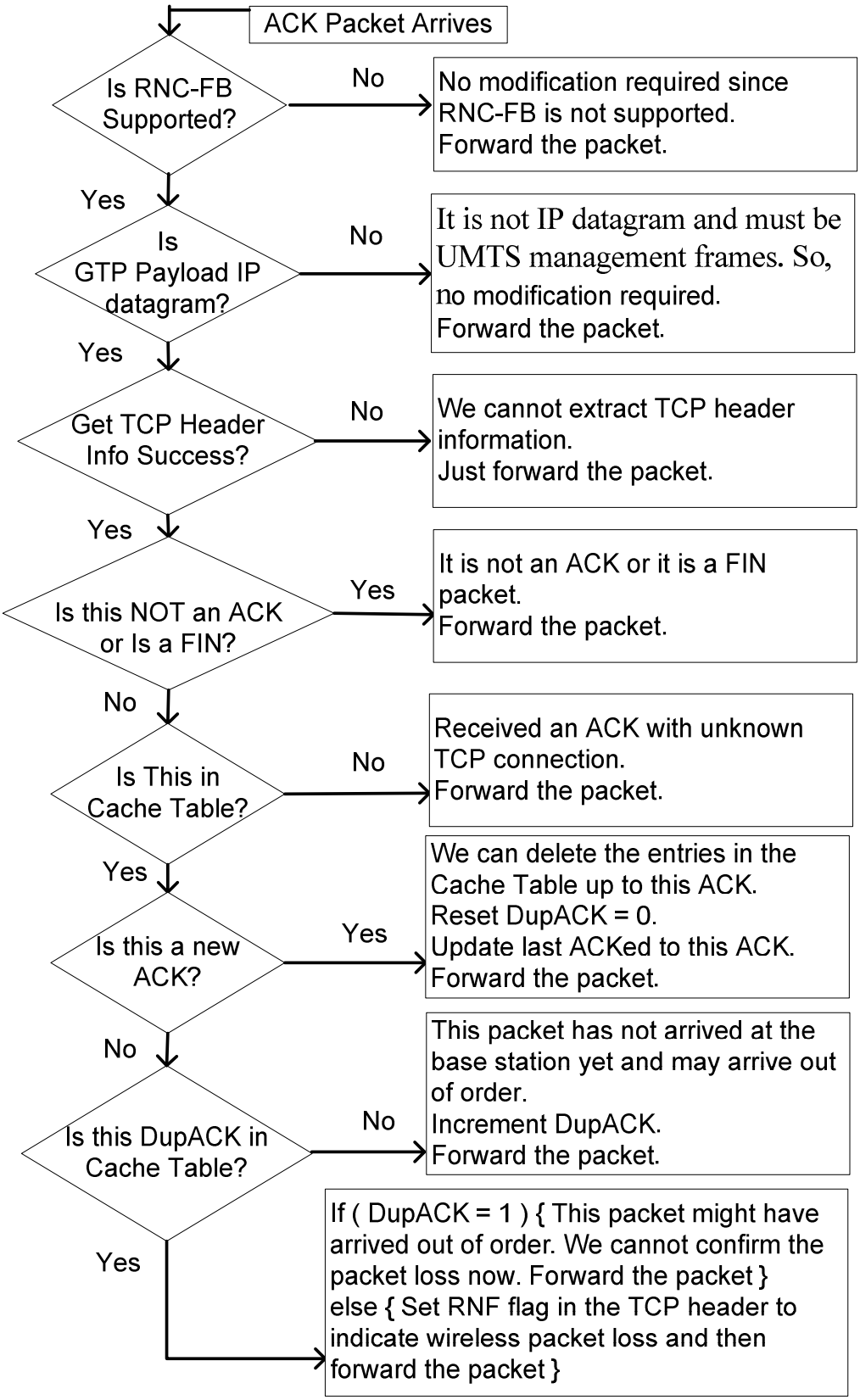

Figure 5-10 Modifications of "GTP ENCAP" state

TCP Reno [RFC 2001, 1997], the current de facto standard for TCP, is fine-tuned to minimize the wireless effects on its performance by adding the RNF flag into TCP header. When TCP sees the ACK with RNF flag set it can confirm that packet is lost and retransmit that packet. However, it is tuned to perform wireless fast retransmit on receiving two duplicate ACKs with RNF flag set because unnecessary TCP retransmissions will contribute to the waste of valuable network resources and significant degradation of TCP end-to-end throughput. Required TCP modifications details are not given here because it is the same as the one implemented in Chapter 4 to enhance the TCP performance over 802.11 WLAN environments.

5.4 SIMULATION RESULTS AND DISCUSSION

To demonstrate the effectiveness of our proposed scheme, the UMTS network model shown in Figure 5.11 is implemented, in turn, with the standard RNC and the modified RNC process model in the RNC protocol stack. The FTP server is configured to generate files of sizes 100 Kbytes and 2 Mbytes. UEs are configured to download 100 Kbyte FTP files except for one UE (UE_1), which downloads 2 Mbyte FTP files and is used for analyzing our simulation results. Modified TCP Reno and UMTS RNC with their default parameters [OPNET Technologies Inc]are used in all simulation scenarios. An extract of the TCP Reno and UMTS RNC parameter values are given in Table 5.1 and 5.2 respectively.

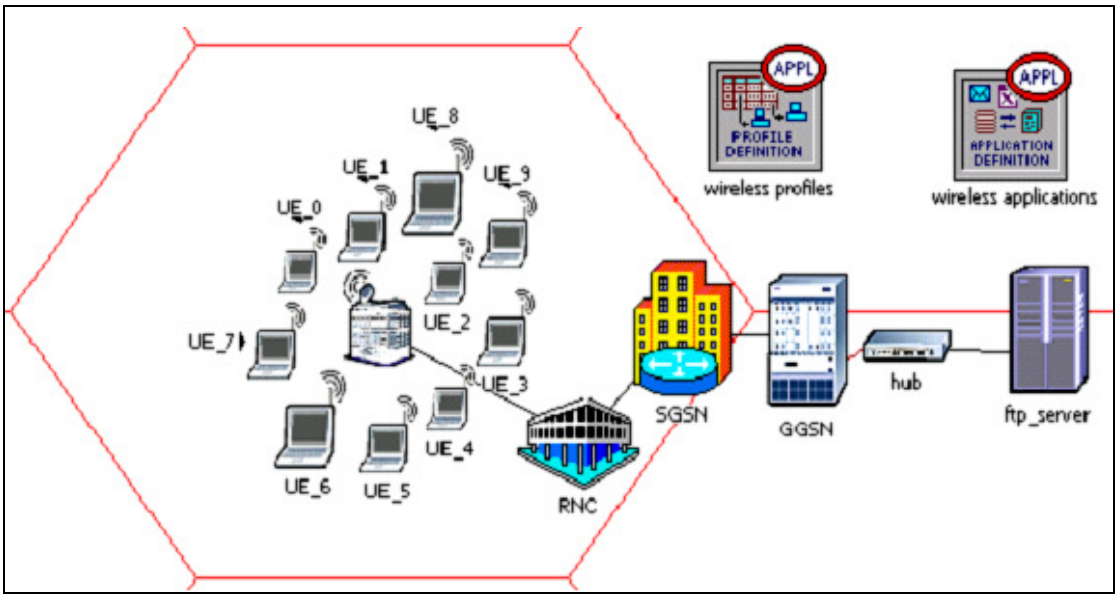

Figure 5-11 UMTS network model

Receive Window Size at FTP Server-1	65535 bytes
Receive Window Size at Mobile Hosts	Default
Delay ACK Mechanism	Segment/Clock Based
Maximum ACK Delay	0.2 seconds
Maximum ACK Segments	2
Maximum Segment Size (MSS)	1460 bytes
Sack Option	Disabled
Slow-Start Initial Count	1 MSS
RTT Gain	0.125
RTT Deviation Gain	0.25
RTT Deviation Coefficient	4
Minimum RTO	1 second
Maximum RTO	64 seconds

Table 5-1 TCP Reno parameters

Transmission Window Size	32
Receiver Window Size	32
SDU Discard Mode	Timer Based No Explicit
Timer MRW	140 milli seconds
Timer Discard	1500 milli seconds
Max MRW	6
Max DAT	4
In-Sequence Delivery	No
UL RLC Mode	Acknowledged Mode
DL RLC Mode	Acknowledged Mode
Admission Control Algorithm	Throughut-Based

Table 5-2 UMTS RNC parameters

Wireless packet drops are generated in user equipments (UEs) using a uniform probability distribution as explained in Figure 4.22 (Section 4.4.2).

5.4.1 SCENARIO 1 RESULTS AND OBSERVATIONS

Extensive simulations were run to obtain the TCP mean throughput value with less than 5% error margin. Figures 5.12 and 5.13 show the number of dropped packets, number of cached TCP headers and the TCP CWND size response of the proposed scheme and that of the standard TCP over UMTS model for packet drop rates of 2% and 5% respectively. Figure 5.14 compares the TCP sent segment sequence number performance of our proposed scheme with that of the standard UMTS for different packet drops rate. In Figures 12 and 13, it is seen that our proposed scheme, with the aid of RNC-FB, significantly increases the TCP congestion window size; it recovers most of the wireless packet losses and minimizes the number of spurious timeouts by early triggering the enhanced wireless fast retransmit and recovery algorithm.

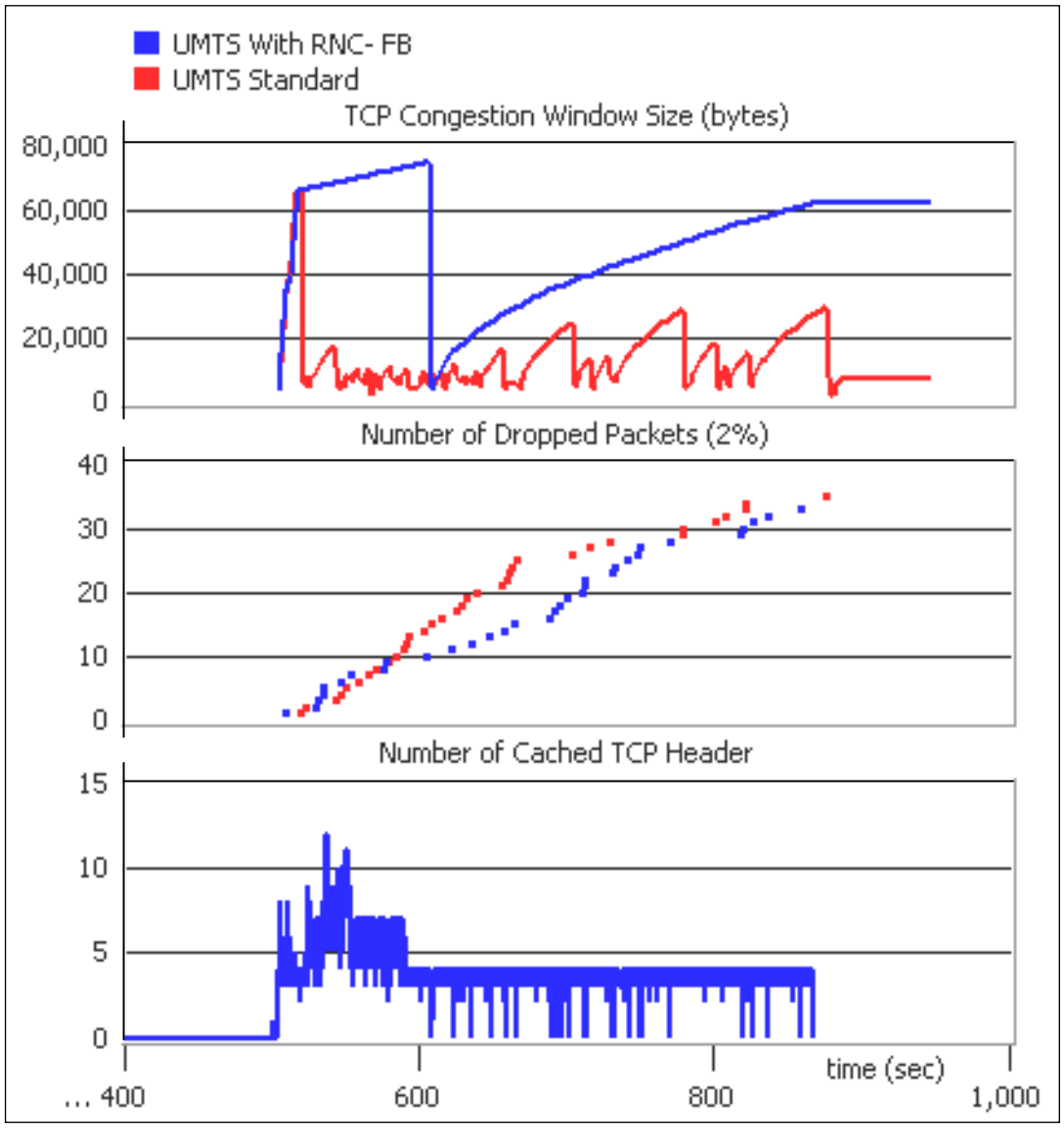

Figure 5-12 TCP CWND, dropped and cached packets

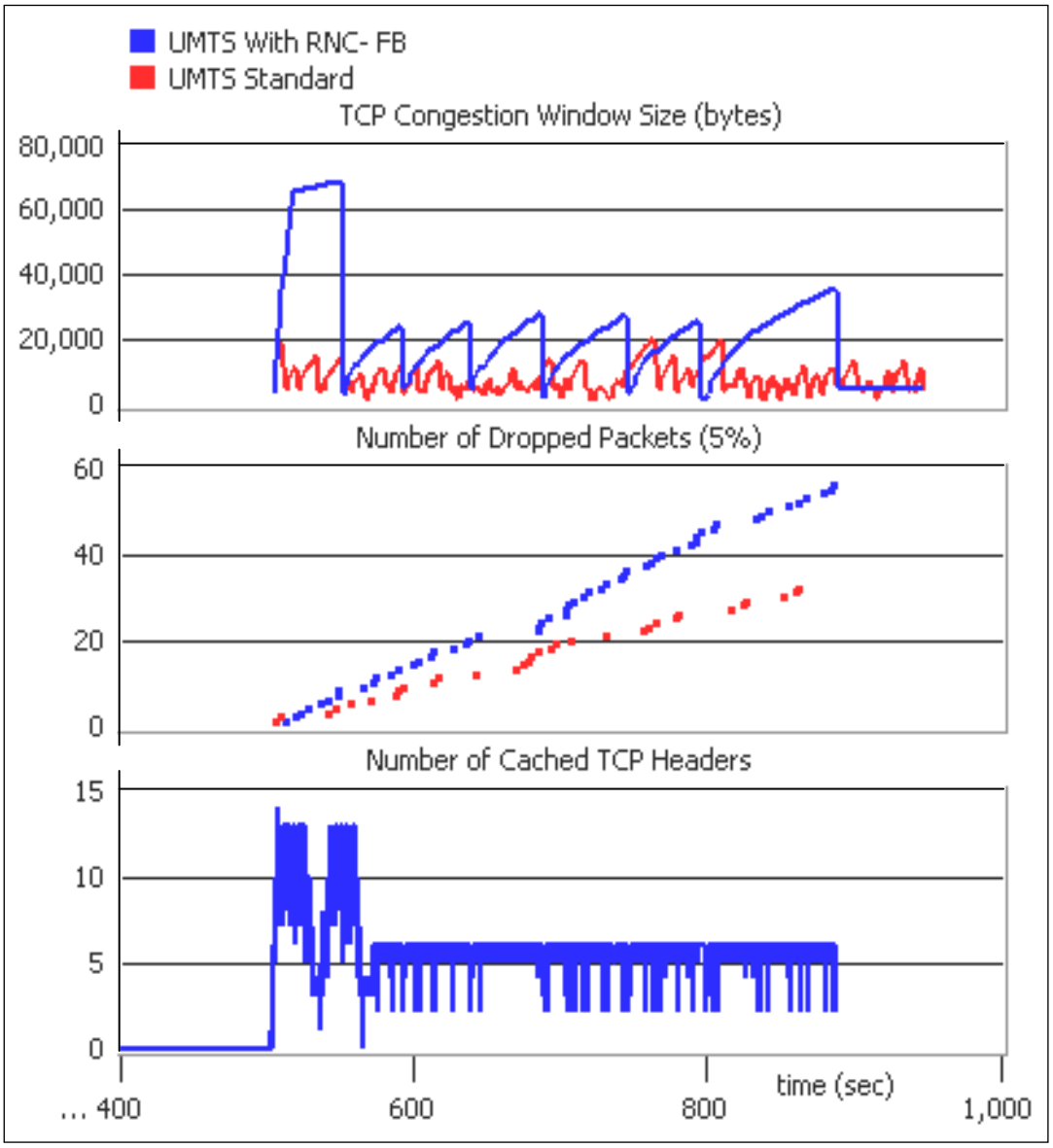

Figure 5-13 TCP CWND, dropped and cached packets

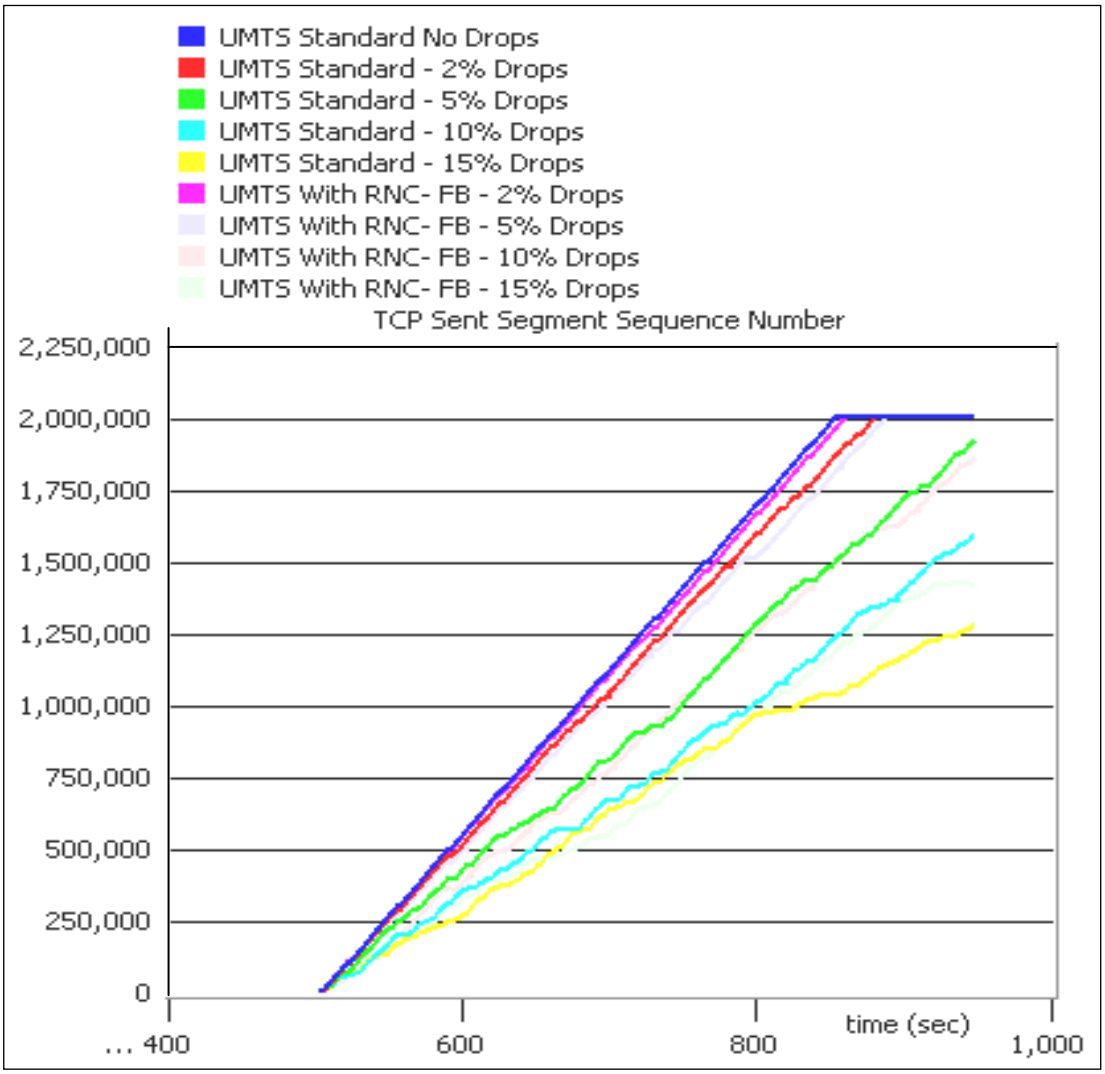

Figure 5-14 TCP sent segment size number

From the TCP performance summary, shown in Table 5.3, it is also observed that our proposed scheme significantly improves the TCP performance with high packet error rates. The number of cached TCP headers, in Figures 5.12 and 5.13, shows that our proposed scheme adds very little overhead to the RNC process model. Since the TCP header information is cached at the RNC, it can also be used to provide additional performance enhancement by freezing the TCP to handle TCP timeouts caused by either handoffs or by temporary wireless disconnections.

Packet drops rate (%)	2	5	10	15
UMTS standard TCP throughput (Kbps)	42.49	24.55	28.48	22.59
UMTS with RNC-FB TCP throughput (Kbps)	44.68	41.55	33.56	25.73
TCP improvement achieved (%)	5.14	20.26	17.81	13.88

Table 5-3 Summary of TCP performance

5.4.2 SCENARIO 2 RESULTS AND OBSERVATIONS

This scenario is designed to see the effects of using TCP Reno with the SACK option enabled on our proposed scheme. Figure 5.15 compares the number of dropped packets, number of cached TCP headers and the TCP CWND size response of our proposed scheme with and without the SACK option enabled for a 10 % packet drop rate. Figure 5.16 compares the TCP sent segment sequence number response of TCP Reno with and without the SACK option enabled for different packet drops rate. In Figure 5.15, it is seen that TCP Reno with SACK option enabled performs much better than does TCP Reno with the SACK option disabled. However, with the packet drop rates of 2% and 10 %, its performance seems to be similar to that with the SACK option disabled. This effect is shown in Figure 5.16.

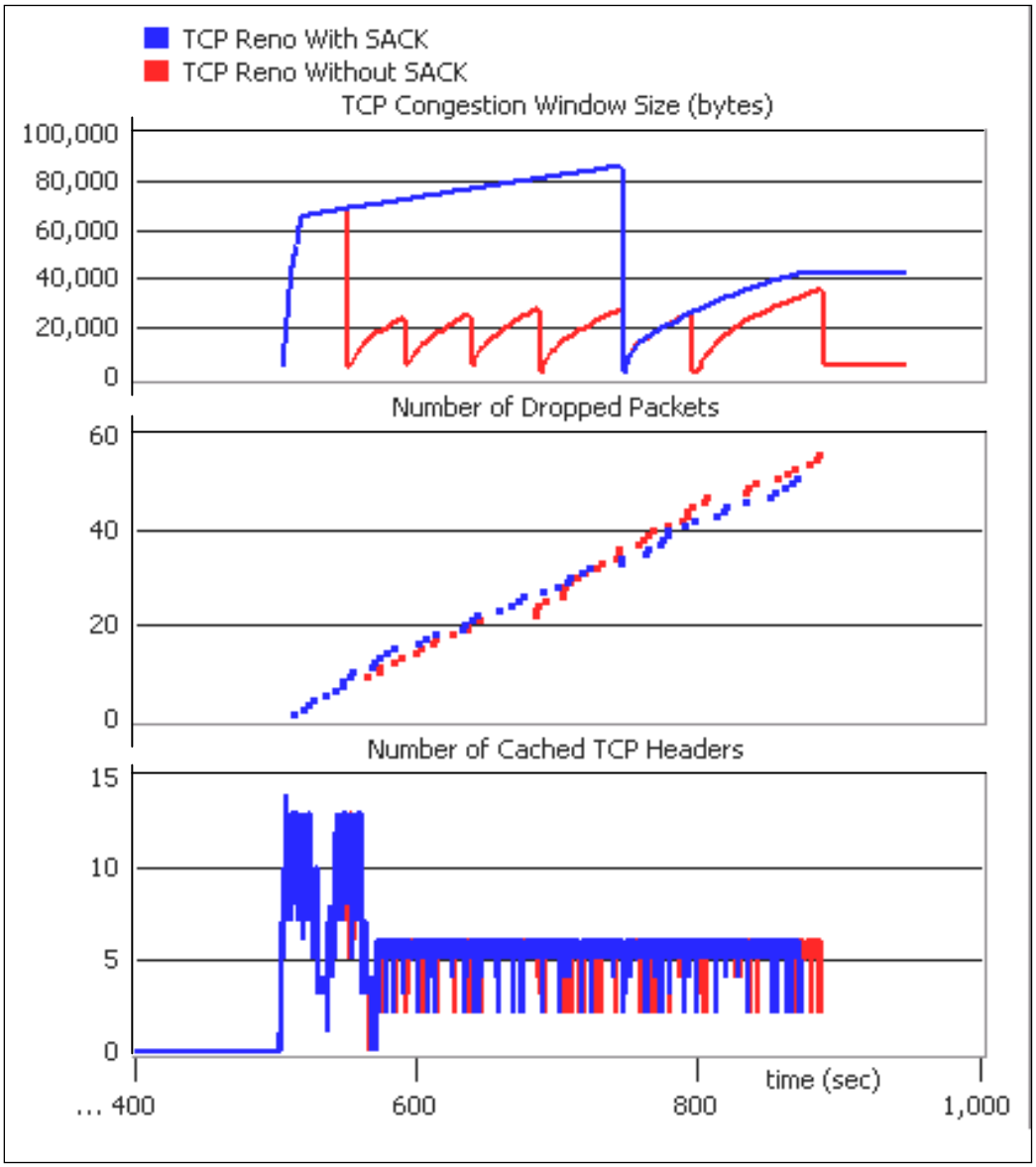

Figure 5-15 TCP CWND, dropped and cached packets

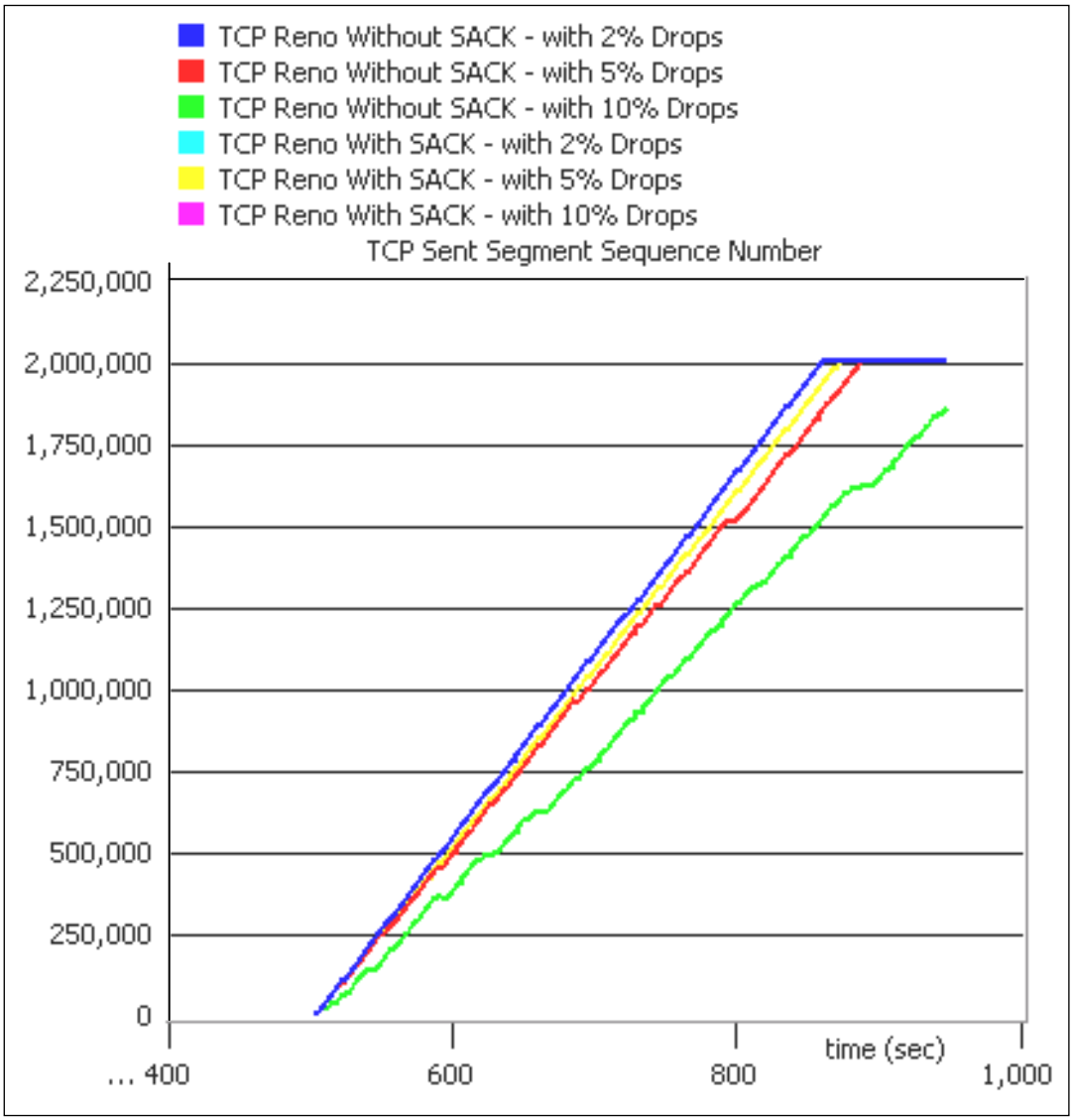

Figure 5-16 TCP sent sequence number

Figure 5.17 shows the total number of RNC down-link packets for WLAN standard, WLAN with RNC-FB with and without the SACK option enabled for different packet drops rate. It can be seen that the RNC-FB without the SACK option better utilizes the wireless resources, compared to that of RNC-FB with the SACK option, with higher packet drops rate. The SACK option will also add additional bytes to the TCP header, thereby reducing the TCP throughput.

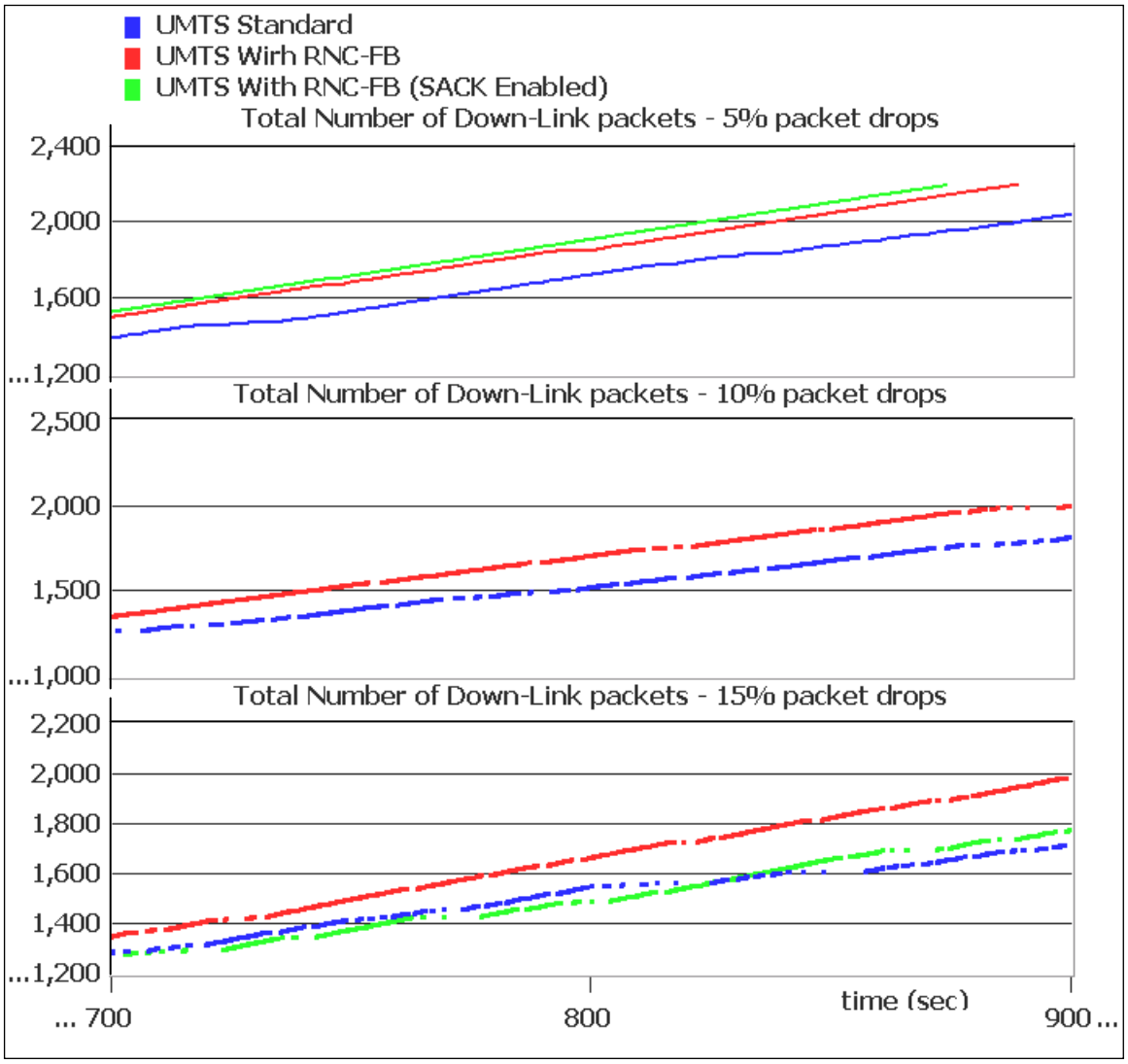

Figure 5-17 Total number of RNC down-link packets

5.5 CONCLUSIONS AND FUTURE WORK

We have proposed and implemented a RNC-FB mechanism in the UMTS network model and run simulation studies to validate the model by comparing its performance with that of the standard TCP over UMTS model. The simulation results showed that the RNC-FB mechanism significantly improved the TCP performance compared to that of standard TCP over UMTS. Specifically, the RNC-FB scheme recovered most of the wireless packet losses and minimized the number of spurious timeouts by early triggering of the enhanced wireless fast retransmit and fast recovery algorithm, introduced in TCP Reno as a modification to accommodate this effect. Utilizing one of the reserved control flags (RNF) enabled the TCP sender to successfully distinguish wireless packet losses from losses due to congestion, thereby avoiding unnecessarily

invocation of the congestion control mechanism, resulting in a higher TCP performance. A minimal modification to the standard TCP is required while maintaining its End-to-End semantics.

The effect of using TCP Reno with the SACK option was also investigated. It was found that our proposed scheme with the TCP Reno SACK option enabled performs better than with the SACK option disabled when the packet drops rate is moderate. Otherwise, RNC-FB without the SACK option better utilizes the network resources.

The size of the cache table required to record TCP headers is quite small and so would not add significant overhead to the RNC process. The TCP header information cached at the RNC can also be used to provide additional performance enhancement by freezing the TCP sender to handle timeouts caused by either handoffs or by temporary wireless disconnections. We are currently investigating the effect of freezing TCP during handoffs on UMTS network performance and also implementing the RNC-FB scheme with other TCP flavors to investigate whether they, too, would benefit from it.

Chapter VI.

TCP Enhancement over Wireless Links by Minimizing Spurious TCP Timeouts

In this Chapter, we analyze the adverse effect on the network performances due to the spurious TCP timeouts and motivate the requirements for improving the TCP congestion control to be able to distinguish spurious timeouts form the traditional timeouts and to behave accordingly. In Section 6.1, we propose a mechanism called wireless enhancement proxy (WENP) that provides radio network feedback to the TCP sender with the aid of two of the TCP header reserved bits. WENP extends our proposed schemes in Chapters 4 and 5 to detect packet loss and large delay across the wireless link and to notify the TCP sender of these events with the aid of two reserved bits in the TCP header. The WENP is implemented in both 802.11 WLAN and UMTS networks and the TCP performance over the WLAN and UMTS networks with and without the wireless timeout detection are explained and compared in Section 6.2 and 6.3, respectively. At last, we draw our conclusions and present the guidelines for further improvement in Section 6.4.

Delay spikes are defined as a sudden and significant change in the RTT between a TCP sender and its receiver. High delay variability has also been observed in fixed wired networks and can be caused, for example, by route flipping [M. Allman & V. Paxson]. In wireless networks on the other hand, the delay variability can be attributed to several factors, most notably the time-varying quality of the wireless link and the hand-off delay. Large and sudden variations in packet transmission delays are often unavoidable in wireless networks. Such large delays are likely to exceed the typical TCP round trip time value. A retransmission timer is a prediction of the upper limit of the RTT. In common TCP implementations, an adaptive retransmission timer accounts for RTT variations [Jacobson, 1988]. A spurious timeout occurs when the RTT suddenly increases to the extent that it exceeds the retransmission timer that had been determined priori. On a spurious timeouts, TCP assumes that all outstanding segments are lost and retransmits them unnecessarily by entering into the slow-start phase.

Spurious TCP timeouts have major impact on congestion control [A. Gurtov & Ludwig]; on one hand, the CWND and the slow-start threshold are reduced unnecessarily after a spurious timeout as no data loss has been yet detected that would otherwise indicate congestion in the network. On the other hand, TCP makes an assumption that all outstanding segments were lost and left the network. In fact, they are likely to be still located in the bottleneck queue. Therefore, Go-back-N retransmissions performed in slow-start phase lead to aggressive sender behavior. That is, while the original transmissions are draining from the queue, the transmission get twice the link rate assuming the receiver generates an ACK for each segment received. This behavior violates the *packet conservation* principle [Jacobson, 1988] and cause real packet loss due to congestion [R. Ludwig & R. H. Katz].

Delay spikes have been observed and measured independently in [Gurtov, Passoja, Aalto, & Raitola, 2002], [Korhonen, Aalto, Gurtov, & Lamanen, 2001] and [Yavuz & Khafizov, 2002]. The effects of large delays and delay variability on the TCP behavior have been investigated in [Shaojian, Atiquzzaman, & Ivancic, 2002] and [A. GURTOV]. Particularly, it is shown that sudden increase in the delay may lead to spurious TCP timeouts that makes the TCP sender enter into slow-start, leading to a low throughput.

TCP Eifel [R. Ludwig & R. H. Katz] solves the retransmission ambiguity by using time stamp option. It can successfully discriminate spurious timeout and normal timeout, but the time stamp option requires additional 12 bytes in the TCP header, resulting in increased overhead in bandwidth constrained wireless networks. In addition, TCP Reno with Eifel experiences performance degradation on the path with sudden delay accompanied by multiple packet losses within one window of data. In this case, resuming the transmission of unsent data can cause multiple timeouts. How to continue with the congestion control when detecting spurious TCP timeouts still remains as research issues.

6.1 PROPOSED WIRELESS TIMEOUT DETECTION SCHEME

We propose a mechanism called wireless enhancement proxy (WENP) that provides radio network feedback, in the form of wireless packet loss and wireless timeout, to the TCP sender. WENP extends our proposed schemes in Chapters 4 and 5 to detect both the packet loss and large delay increase across the wireless link and to notify the TCP sender of them with the aid of two reserved bits, called wireless loss notification (WLN) and wireless timeout notification (WTN), in the TCP header, as illustrated in Figure 6.1.

Source port number (16-bits)												Destination port number (16-bits)
Sequence number												
Acknowledgement number												
4-bit Header length	Reserved bits	WTN	WLN	WCR	ECE	URG	ACK	PSH	RST	SYN	FIN	Receiver window size (16-bits)
Checksum (16-bits)												Urgent Data (16-bits)
Options												

Figure 6-1 TCP header with WLN and WTN reserve bits

Figure 6.2 shows the WENP process model. The *MONITOR DATA* state and *MONITOR ACK* state are used to record the arrival times of data packets and their corresponding ACKs at the base station in order to measure the wireless round trip time (W-RTT) and to detect wireless timeouts with the aid of a timer, similar to the standard TCP retransmission timer, which times out when acknowledgements are not received in time across the wireless link. WENP modifies the TCP Connection Table data structure and the Global structure for Cache Table defined in Chapter 4 and Chapter 5, as shown in Figures 6.3 and 6.4 respectively, to be able to detect both the wireless packet loss and large delay across the wireless link.

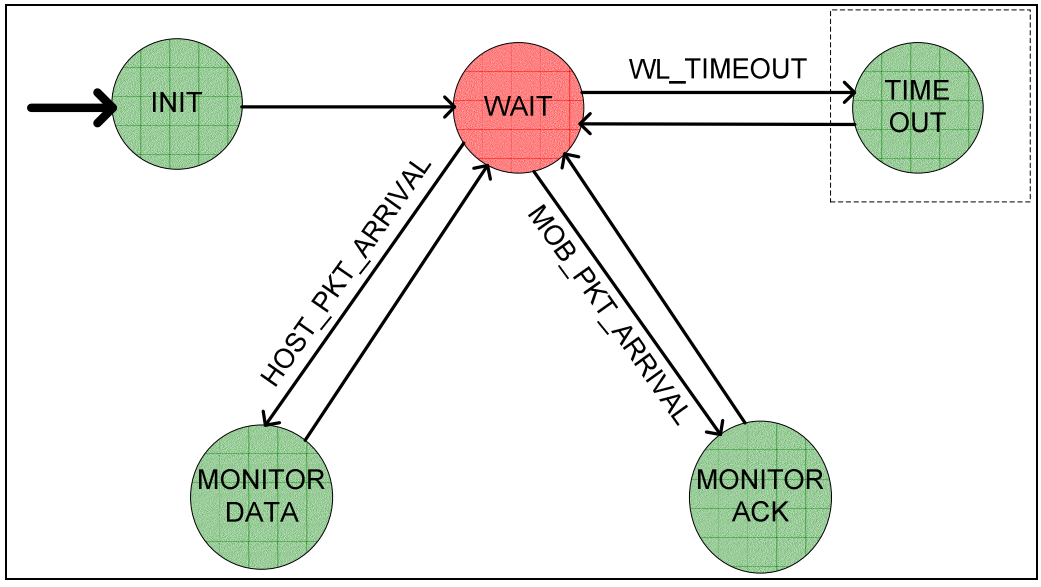

Figure 6-2 WENP process model

```
typedef struct
{
      unsigned   int     src_ip;
      unsigned   int     dest_ip;
      int                src_port;
      int                dest_port;
      unsigned   int     last_seq_num;
      unsigned   int     last_ack_num;
      int                repeat_ack;
      int                fin_flag;
      unsigned           fin_seq_num;
      Evhandle           timeout_evt;
      TcpInfo           *lastAckedPt;
      int                time_out_active;
      int                wireless_loss;
      double             wireless_rtt;
      double             wireless_mean_rtt;
      double             wireless_time_out_value;
      double             wireless_mean_rtt_dev;
      int                get_wrtt_enable;
      int                time_out_count;
} struct_TcpConnectionTable;
```

Figure 6-3 TCP Connection Table Data Structure

```
typedef struct
{
      TcpInfo      *tcp_infoPt[WLD_CACHE_SIZE];
      int          ici_addr[WLD_CACHE_SIZE];
      int          max_cached;   /* Maximum number of packet header that can be cached at
                                    the same time */
      int          num_cached;   /* Total number of packets header that has been cached */
      int          num_removed;  /* Total number of packets header that has been removed */
      int          curr_cached;  /* Current number of packets header cached */
      double       time_rcvd[WLD_CACHE_SIZE];
} struct_TcpHeaderCache;
```

Figure 6-4 Global Structure for Cache Table

Note that the variable time_rcvd in the Global structure for Cache Table holds the actual time the data packet received and is used to measure the W-RTT on receiving an ACK for that packet.

WENP measures the W-RTT only if a packet is successfully acknowledged, i.e., on receiving a new ACK, as the standard TCP does. The *MONITOR ACK* state is designed to calculate the wireless RTT (W-RTT) and the wireless RTO (W-RTO) as explained in the following Section.

6.1.1 WIRELESS RTT MEASUREMENT

A pseudo code for measuring W-RTT is shown in Figure 6.5. W-RTT measurement and the update of wireless timeout value is based on Karn's algorithm [P. Karn & C. Partridge, 1995]. Note that Karn's algorithm restricts retransmission timeout updates for retransmitted segments in order to avoid retransmission ambiguity. The reason is that if the RTT measurement is based on the actual transmission time of the original packet, the RTT estimate may be too pessimistic. If the RTT measurement is based on the transmission time of the most recent retransmitted packet may result in too pessimistic estimate [Fu & Atiquzzaman, 2005]. WENP restricts W-RTT measurement not only for the retransmitted packet but also for the packets that are being acknowledged after a packet loss recovery by utilizing get_wrtt_enable variable that is set as shown in Figure 6.6.

```
get_wireless_RTT()
{
  if(get_wrtt_enable)
    {
      wireless_rtt = current_time – time_in_which_packet_received;
      rtt_error =  wireless_rtt – wireless_mean_rtt;
      wireless_mean_rtt += wireless_rtt_gain * rtt_error;
      wireless_mean_rtt_dev +=  wireless_rtt_dev_gain *
                                  (fabs(rtt_error) – wireless_mean_rtt_dev);
      wireless_time_out_value = wireless_mean_rtt +
                                  wireless_rtt_dev_coef * wireless_mean_rtt_dev;
    }
}
```

Figure 6-5 A pseudo code for measuring W-RTT

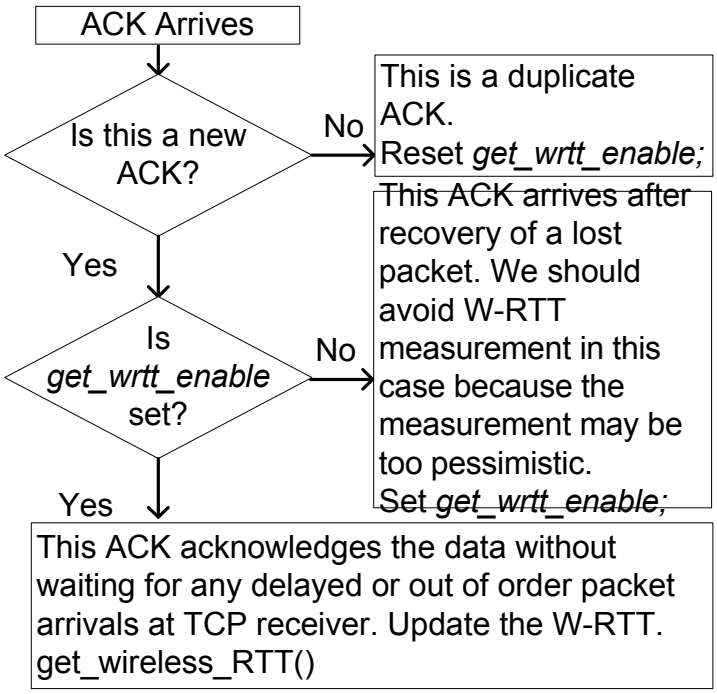

Figure 6-6 Setting get_wrtt_enable variable

On receiving a data packet from the TCP sender, WENP caches the TCP header, updates the TCP Connection Table and sets a timer for this data packet, with the timeout value obtained as shown in Figure 6.5. The timer expires when an excessive delay is experienced over the wireless link. When this timer expires, WENP enters into the *TIMEOUT* state that sends a duplicate ACK packet to the TCP sender with WTN flag set and backs off the timer as the standard TCP does. The feedback ACK packet with the WTN flag set enables the TCP sender to distinguish spurious timeouts from normal timeouts. Note that the TCP Connection Table Data Structure holds the last acknowledged packet for each TCP connection. In the event of a wireless timeout, this ACK packet is used by the *TIMEOUT* state to prepare and send a feedback packet.

The RTT across the wireless link, which has just one hop, will be short and retransmissions due to packet corruption make W-RTT fluctuate considerably. It is important not to inject many feedback packets into the network because this may cause a negative impact on the TCP performance. In fact, WENP does not inject any additional packet into the network when it detects wireless packet loses, rather it sets the WLN flag. In the case of wireless timeout detection, WENP does inject feedback packets into the network but limited to only one packet per a window of data.

6.1.2 REQUIRED TCP CONGESTION CONTROL MODIFICATIONS

In a standard TCP implementation, the TCP sender concludes that the network has dropped a packet when it receives three duplicate ACKs. It then immediately retransmits the dropped packet in order to avoid the expiry of the retransmission timer. The rationale is that the sooner the fast retransmit occurs, the better TCP performs because it avoids unnecessary TCP timeouts. WENP enables the TCP sender to confirm a wireless packet loss when it receives an acknowledgement with the WLN flag set. As proposed in Chapters 4 and 5, the standard TCP is fine tuned to retransmit the packets dropped across the wireless link without reducing the congestion window when it receives just two duplicate ACKs with the WLN flag set.

Another new feature added to the standard TCP enables the system to distinguish spurious timeouts from normal timeouts. Recall that WENP utilizes the last acknowledged packet that was cached in its Connection Table to send a feedback packet to the TCP sender with WTN flag set when its timer expires, indicating a large delay across the wireless medium. When the timer expires, WENP sets the WTN flag only if it determines that it has received packets form the TCP sender since it has last acknowledged a packet. This confirms that no timeout occurred in the wireline part of the network and so avoids misinterpreting a normal timeout as a wireless timeout. Now, if the TCP sender receives duplicate ACK with the WTN flag set, it should consider the following cases to decide how to proceed with the transmission:

- If both WTN and WLN flags are set, it indicates that TCP sender has already received a duplicate ACK with only the WLN flag set. Now it has received the second duplicate ACK as a result of wireless timeout. It implies that there is a packet flow exists across the wireless link, but has experienced a sudden large delay. This is the cause for spurious TCP timeout. In this case, TCP sender is modified to trigger wireless fast retransmit without reducing the congestion window.

- If only the WTN flag is set, it indicates that the wireless link is congested. In this case, it is highly possible that multiple packets are lost across the wireless link or even at the base station due to buffer overflow. Considering the wireless link is also a part of the network, we treat this case as congestion in the network and leave TCP to handle it using its standard fast retransmit and recovery mechanism or its timeout recovery mechanism, depending on the way the packet loss is detected.

6.2 EXPERIMENT - 1: 802.11 WLAN NETWORK WITH THE PROPOSED SCHEME

The network model used for this study is shown in Figure 6.7. The LAN is extended using a WLAN Ethernet router that forms a WLAN together with some mobile hosts (MHs).

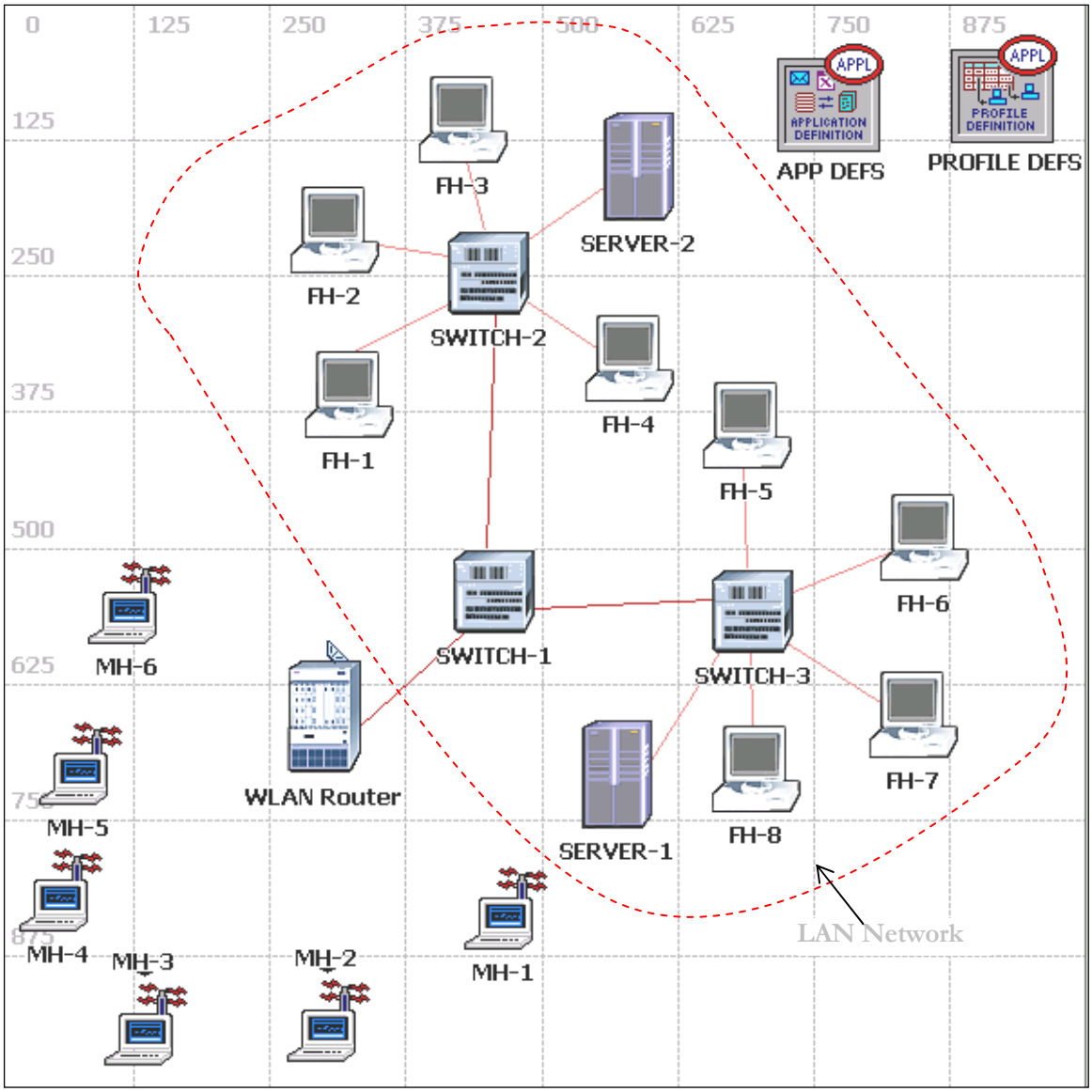

Figure 6-7 WLAN network model

The model consists of two Servers and some fixed hosts and mobile hosts. Servers and fixed hosts are connected to the WLAN router through switches using 10baseT point-to-point link model. Servers are equipped with modified TCP Reno while the WENP is introduced between the MAC and IP layer of the WLAN Router. The Application and the Profile Configuration nodes are configured to generate different applications such as HTTP, FTP Database and Email.

Fixed hosts are configured to utilize some of these services in parallel with MHs in order to make the network analyzes be realistic.

All MHs are configured to download 1.6 MByte FTP files simultaneously with different packet drops rate as shown in Table 6.1. All MHs download FTP files from Server 1 while the FHs download different applications from both Servers 1 and 2. Data Packets coming from FTP Servers are dropped in MHs, at the MAC layer, using a uniform probability distribution. The MAC layer is modified to generate bursty packet drops, as illustrated in Figure 4.22, so that the base station will perform its local retransmission and discard the packets once the threshold number of retry limit reaches. The packets that experience base station local retransmission before get successfully transmitted will imitate characteristics of wireless links, thereby impacting the End-to-End RTT.

Mobile Hosts	MH-1	MH-2	MH-3	MH-4	MH-5	MH-6
Packet drops rate(%)	0	5	10	15	20	25

Table 6-1 MHs configurations

The WLAN Router is implemented, in turn, with a standard WLAN, a WLAN with Snoop and a WLAN with WENP to compare their relative performances. Modified TCP Reno with the default parameters is used in all simulation scenarios. Selected WLAN and TCP Reno parameter values are given in Tables 6.2 and 6.3, respectively.

Physical Characteristics	Direct Sequence
Data Rate	11 Mbps
Transmit Power	0.005 W
Packet Reception-Power Threshold	-95 dBm
Fragmentation Threshold	None
Short Retry Limit	7
Long Retry Limit	4
Max Receive Lifetime	0.5 seconds
Buffer Size	256000 bits
PCF Parameters	Disabled
Time Stamp	Disabled
BSS Identifier	Auto Assigned

Table 6-2 WLAN parameters

Maximum Segment Size (MSS)	1460 bytes
Receive Window Size at Mobile Hosts	Default
Delayed ACK Mechanism	Segment/Clock Based
Maximum ACK Delay	0.2 seconds
Maximum ACK Segments	2
Slow-Start Initial Count	1 MSS
Duplicate ACK Threshold	3
Fast Retransmit	Enabled
Fast Recovery	Reno
Selective ACK (SACK)	Disabled
Time Stamp	Disabled
Initial RTO	3.0 seconds
RTT Gain	0.125
Deviation Gain	0.25
RTT Deviation Coefficient	4.0

Table 6-3 TCP parameters

6.2.1 SIMULATION RESULTS AND DISCUSSION

Figure 6.8 shows TCP CWND size, number of MAC packets dropped and number of cached TCP headers while Figure 6.9 shows the W-RTT measurements during FTP file upload, with different packet drops rate, with the proposed scheme. From Figure 6.8, it can be seen that WENP adds little overhead to the WLAN Router and has successfully recovered from most of the wireless packet losses and minimized the number of spurious timeouts by early triggering the enhanced wireless fast retransmit and fast recovery algorithm. However, it can be observed that the WENP experienced a few TCP timeouts with higher packet drops rate. It can be attributed to the TCP Reno's inability to recover form multiple losses within a window of data. It should also

be noted that if a retransmitted packet is dropped, it can only be recovered by the TCP timeout process.

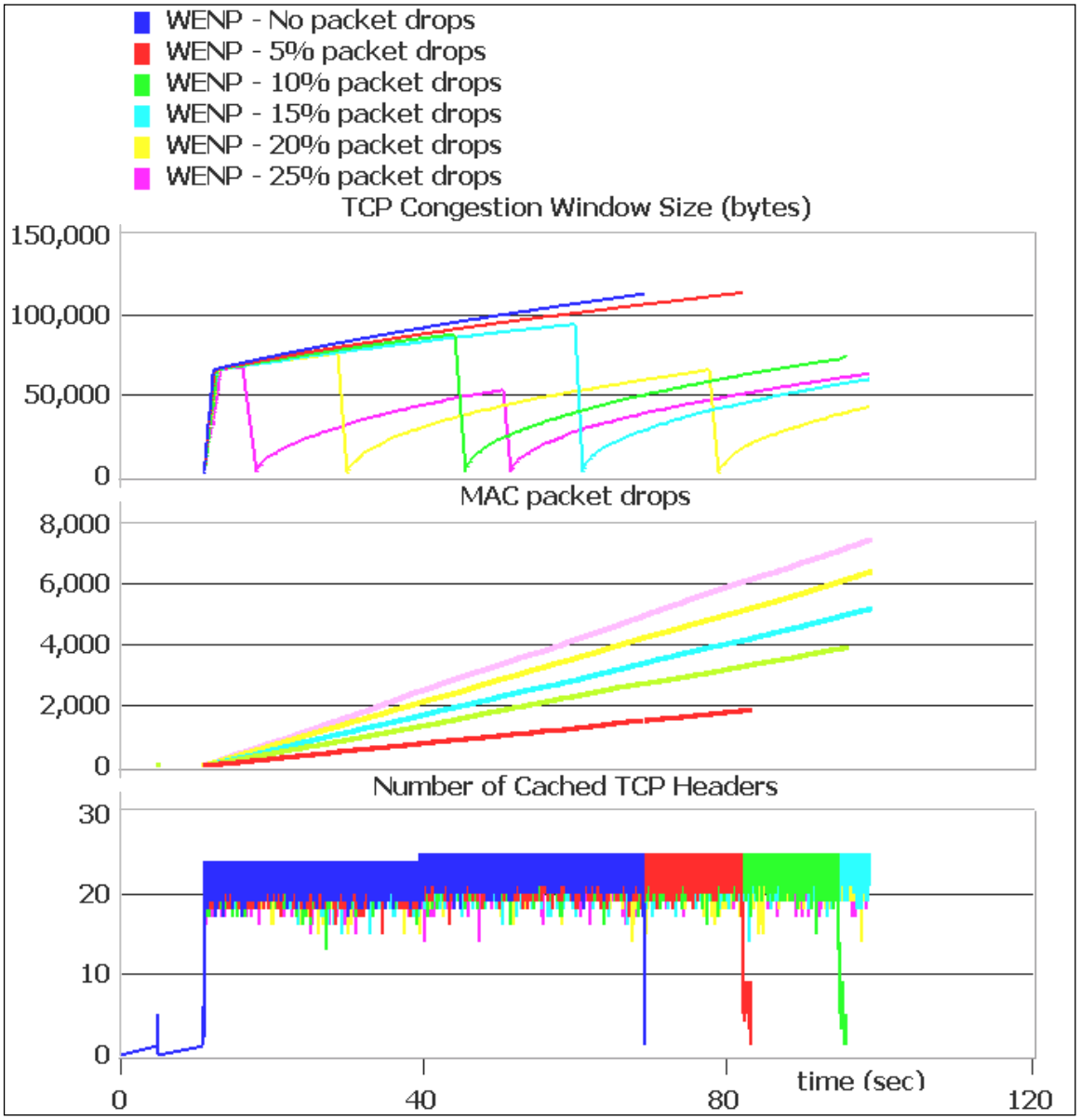

Figure 6-8Responses during MH-3 FTP file upload

As expected, W-RTT measurement, shown in Figure 6.9, was obtained using the cached table seems to fluctuate. However, it helps WENP to detect wireless timeouts and enables the system to minimize the spurious TCP timeouts, thereby further improving the TCP performance.

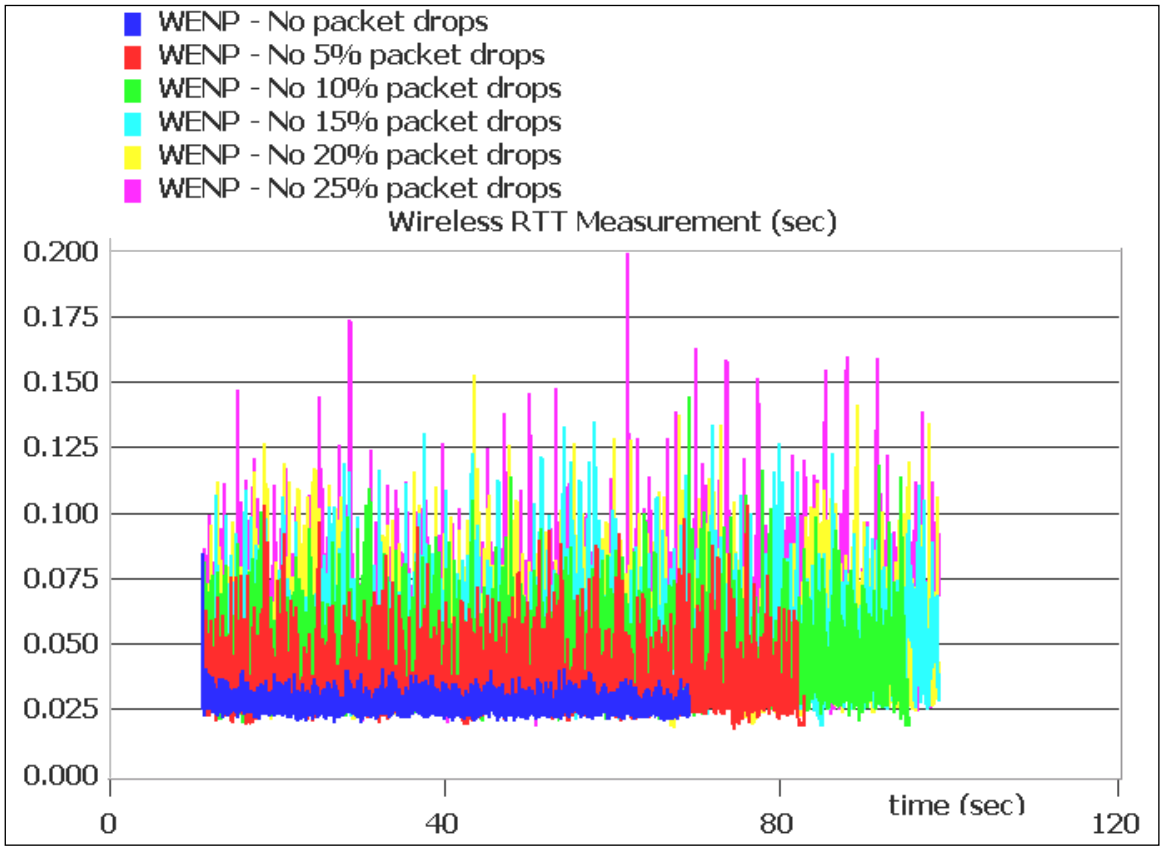

Figure 6-9 W-RTT measurement

Notice that WENP is designed to improve the TCP performance without changing its original behavior. For example, TCP Reno cannot recover from multiple packet losses within a window of data. This effect can be observed in Figures 6.8.

Figures 6.10 and 6.11 compare the TCP CWND history and TCP sent segment sequence number responses, respectively, for the different cases. It is seen that the proposed scheme significantly improves the TCP performance in comparison with the standard WLAN both without and with the Snoop protocol.

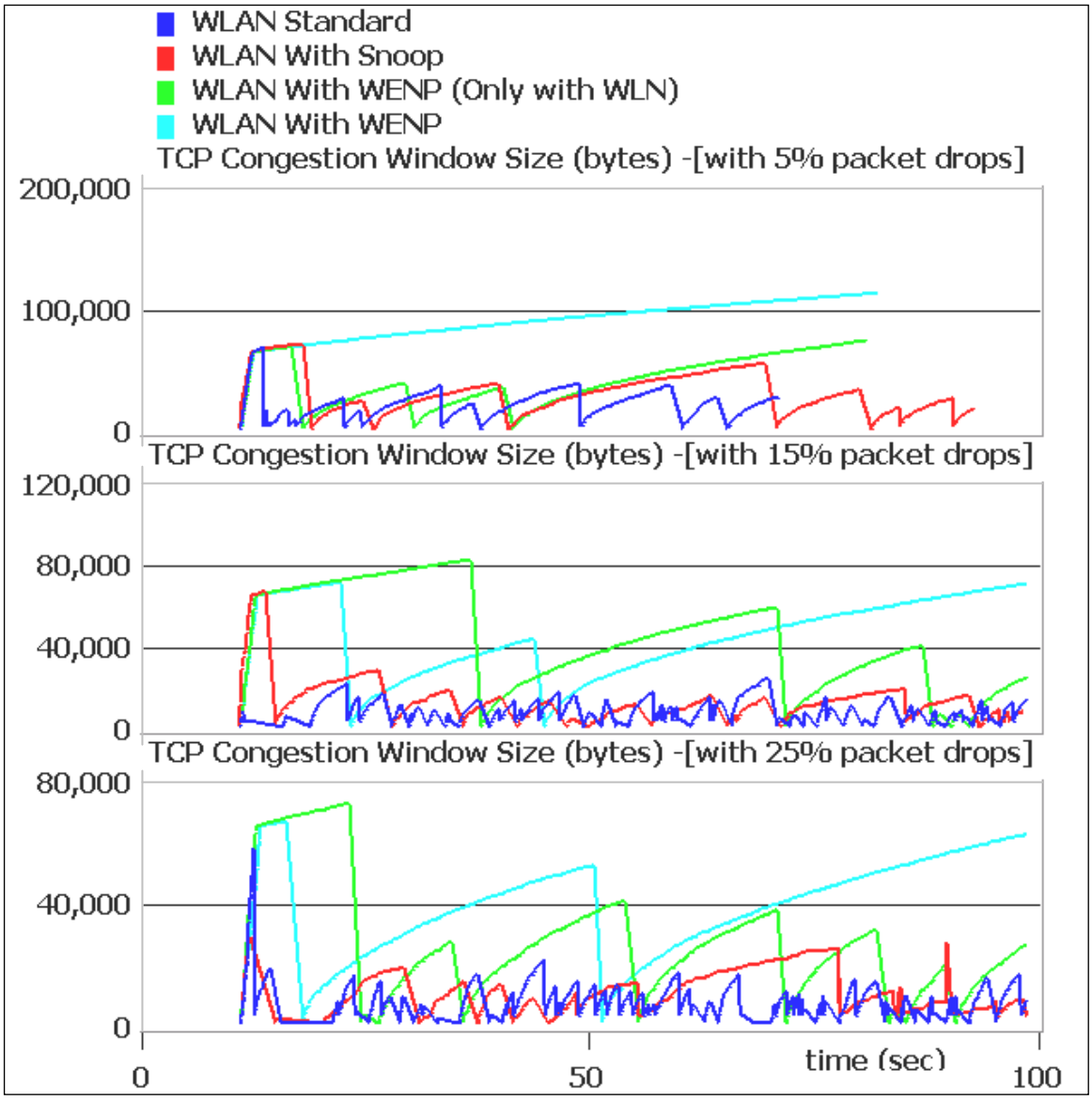

Figure 6-10 TCP CWND size responses

The WENP implementation with only wireless packet loss detection is also included to show the effects of wireless timeouts on TCP performance. From Figures 6.10, it can be observed that TCP CWND size response of WENP without wireless timeout detection experiences more timeouts with higher MAC packet drops rate. However, WENP equipped with the timer helps TCP to minimize unnecessary spurious TCP timeouts, thus providing further enhancement to TCP performance.

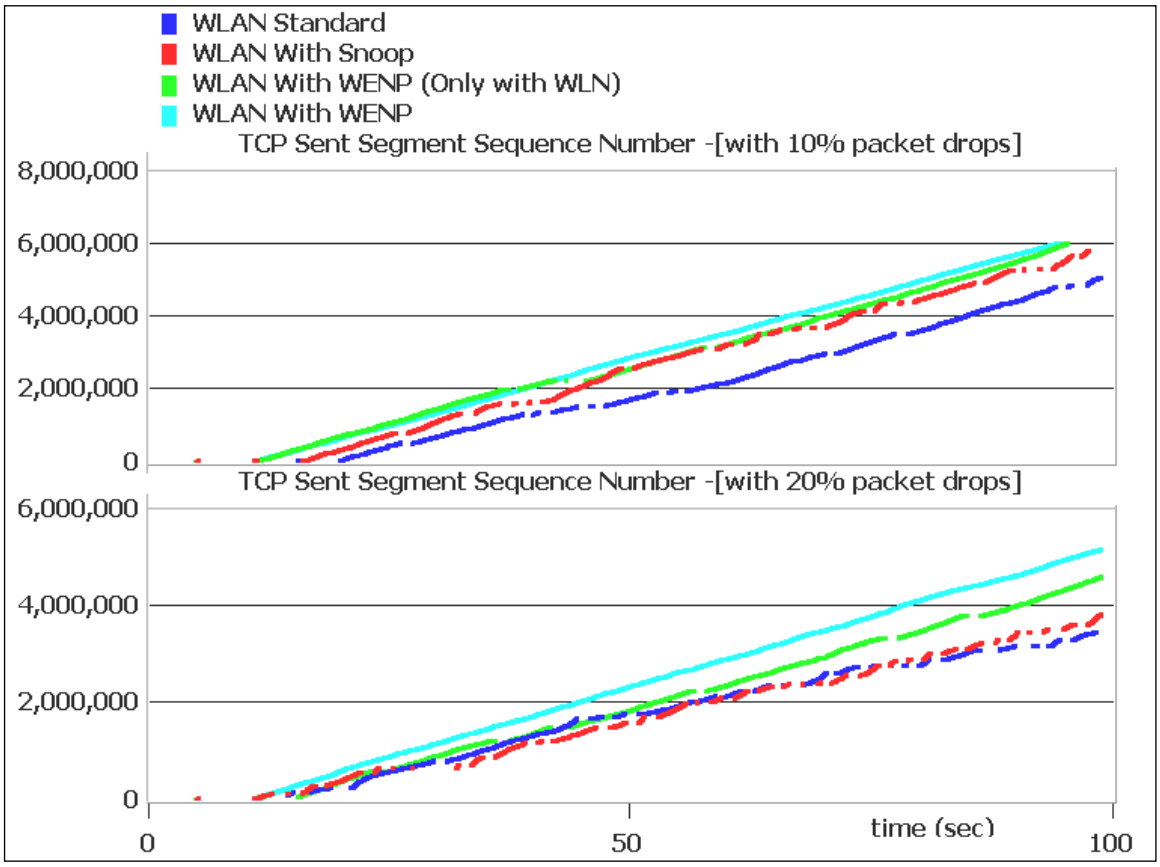

Figure 6-11 TCP Sent Segment Sequence Number responses

The average WLAN throughput and WLAN delay responses shown in Figure 6.12 indicates that WENP has efficiently utilized the available wireless resources by adapting to the wireless channel characteristics. It should also be noted that the Snoop performance was degraded. This can be attributed to Snoop's inability to synchronize with the TCP sender in order to be able to avoid unnecessary spurious TCP timeouts and competing for retransmissions. WENP adds much less overhead to the base station than Snoop does because it only caches the TCP header information. It thereby improves the WLAN delay response as well.

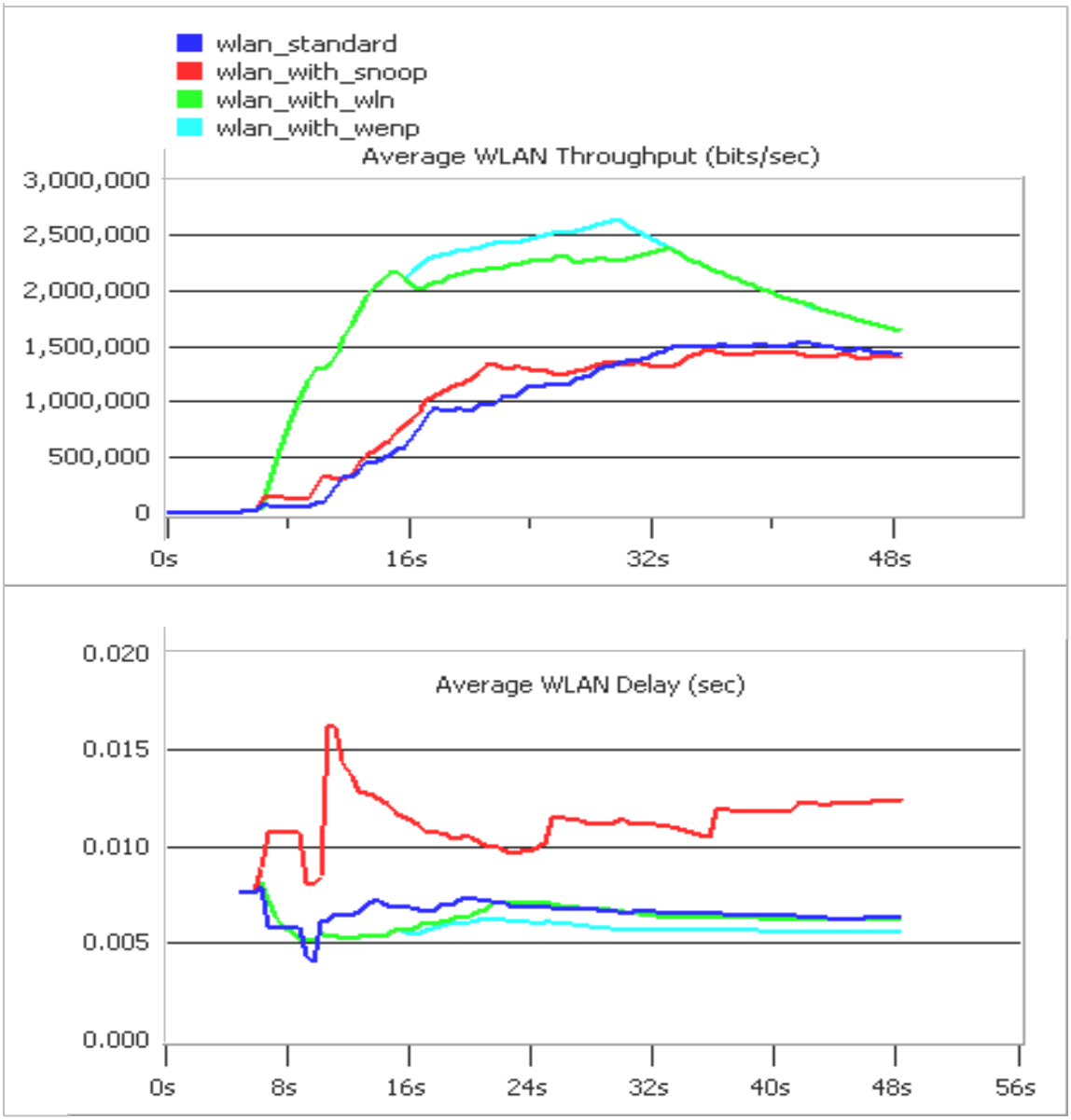

Figure 6-12 WLAN throughput and Dealy (sec) responses

Table 6.4 summarizes the total TCP throughput performance during the FTP file upload with different packet drops rate with 5% error margin. Notice that we run this simulation, by allowing all mobile hosts to drop MAC packets with the same rate, in order to see how the proposed scheme utilizes the available network resources. From Figures 6.10 and 6.11, and Table 6.4, it can be seen that the proposed scheme recovered from wireless effects by early triggering of the enhanced fast retransmit and recovery mechanism, better utilized the available network resources and has dramatically increased the TCP throughput compared to that of the standard WLAN, both without and with the Snoop protocol.

Packet drops rate (%)	Total TCP throughput (Mbits/sec)			Improvement compared to WLAN with Snoop(%)	Improvement compared to Standard WLAN (%)
	Standard WLAN	WLAN with Snoop	WLAN with WENP		
0	4.96	4.96	4.96	0	0
5	3.11	3.52	4.14	17.6	33.1
10	2.35	2.78	3.44	23.7	46.4
15	1.78	2.16	2.92	35.2	64.1
20	1.45	1.78	2.69	51.1	85.5
25	1.29	1.58	2.51	58.9	94.6

Table 6-4 Summary of TCP throughput performacne

6.2.2 CONCLUSIONS

A new scheme, WENP, was presented that detects and distinguishes wireless packet losses and wireless timeouts from congestion related packet losses and timeouts. WENP was implemented in a WLAN model in OPNET with modified TCP Reno as the transport protocol. Simulation results showed that WENP improved the TCP performance significantly compared to that of both the standard WLAN and a WLAN with Snoop. It enabled the modified TCP to trigger enhanced wireless fast retransmit and fast recovery mechanisms to recover from wireless packet losses sooner using the WLN flag.

WENP also implemented a timer that detects wireless timeouts and enables the TCP sender to avoid spurious TCP timeouts with the aid of both the WTN and WLN flags, further enhancing the TCP performance. Simulation results also showed that WENP can handle multiple TCP connections and utilized the available network resources efficiently by adapting to the network characteristics. WENP does not inject any additional packet into the network to provide feedback when it detects wireless packet loses; it only sets the WLN flag. In case of wireless timeout detection, WENP does inject feedback packets into the network but limited to only one packet per a window of data.

6.3 EXPERIMENT 2: UMTS NETWORK WITH THE PROPOSED SCHEME

The proposed scheme WENP is implemented in UMTS RNC protocol stack as described in Section 5.3. Specifically, the GTP layer in the RNC protocol stack is equipped with WENP, as shown in Figure 6.13, to detect and notify the TCP sender of any wireless loss or wireless timeouts. The detection of wireless packet losses and wireless timeouts are the same as explained in Section 6.1.

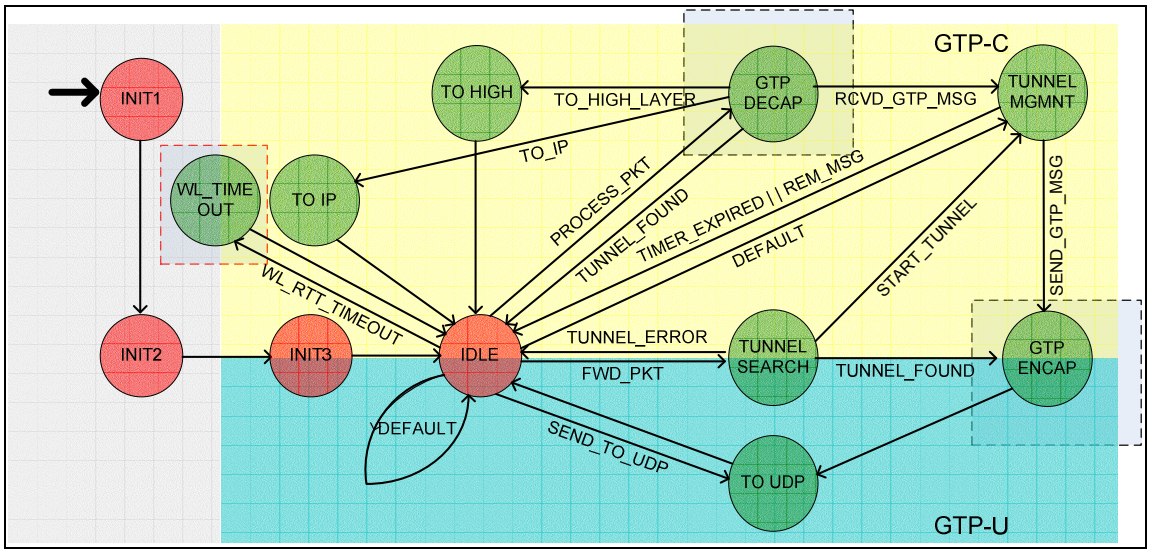

Figure 6-13 GTP process model with WENP

To demonstrate the effectiveness of our proposed scheme, the UMTS network model shown in Figure 6.14 is implemented, in turn, with the standard RNC and the modified RNC process model in the RNC protocol stack. The FTP server is configured to generate FTP files of 100 Kbytes, 500 Kbytes and 1 Mbytes. User equipments are configured to download FTP files with packet drops, generated using a uniform probability distribution function, as described in Table 6.5. Modified TCP Reno and UMTS with their default parameters [OPNET Technologies Inc] are used in all simulation scenarios. Selected UMTS RNC parameter values are given in Table 6.6.

User Equipments	1	2	3	4	5	6	7	8	9	10	11
Packet drops rate(%)	0	2	4	6	8	10	12	14	16	18	20
FTP file size (Mbyts)	1	1	1	1	0.5	0.5	0.5	0.1	0.1	0.1	0.1

Table 6-5 Mobile host configurations

Transmission Window Size	32
Receiver Window Size	32
SDU Discard Mode	Timer Based No Explicit
Timer MRW	140 milli seconds
Timer Discard	1500 milli seconds
Max MRW	6
Max DAT	4
In-Sequence Delivery	No
UL RLC Mode	Acknowledged Mode
DL RLC Mode	Acknowledged Mode
Admission Control Algorithm	Throughut-Based

Table 6-6 Selection of UMTS parameters

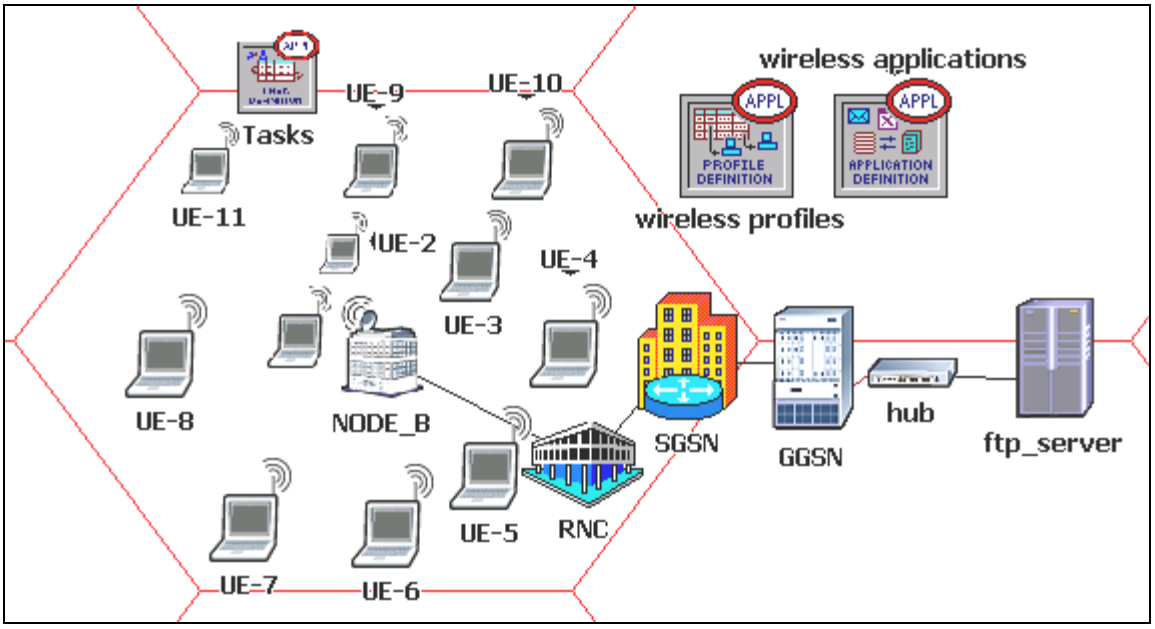

Figure 6-14UMTS network model

6.3.1 SIMULATION RESULTS AND DISCUSSION

Extensive simulations were run to get the mean TCP throughput with less than 5% error margin. Figures 6.15 shows the results of W-RTT measurement, number of cached TCP headers, number of dropped packets, and the TCP CWND size response of our proposed scheme and that of the standard TCP over UMTS model for packet drop rates of 6%. Notice that our scheme without the wireless timeout detection is also included to show the effect of spurious TCP timeouts. It can be observed that the proposed scheme has quickly recovered from both wireless loss and wireless timeouts; it has experienced only one TCP timeout and it can be attributed to the standard behavior of TCP Reno itself. As expected, one can see considerable fluctuations of WRTT. However, it helps WENP to detect wireless timeouts and enables the system to minimize spurious TCP timeouts, thereby further improving the TCP performance.

It can also be seen that there is a maximum of 16 TCP header information items cached during the transmission period, which adds very little overhead to the RNC process model. Since the TCP header information is cached at the RNC, it can also be used to provide additional performance enhancement by freezing TCP to handle TCP timeouts caused by handoffs.

Figures 6.16 compares the TCP CWND size responses of the proposed scheme with that of the standard UMTS for different packet drops rate. It can be seen that the proposed scheme, WENP, significantly increases the TCP CWND size; it has recovered most of the wireless packet losses and has reduced the number of spurious timeouts by early triggering the enhanced wireless fast retransmit and recovery algorithm. However, it can be noted that even with the proposed scheme, TCP timeouts occurred at higher packet drops rate. This can be attributed to the TCP Reno's inability to recover from multiple losses within a window of data. It should be noted that if a retransmitted packet is dropped, it can only be recovered by the TCP timeout process.

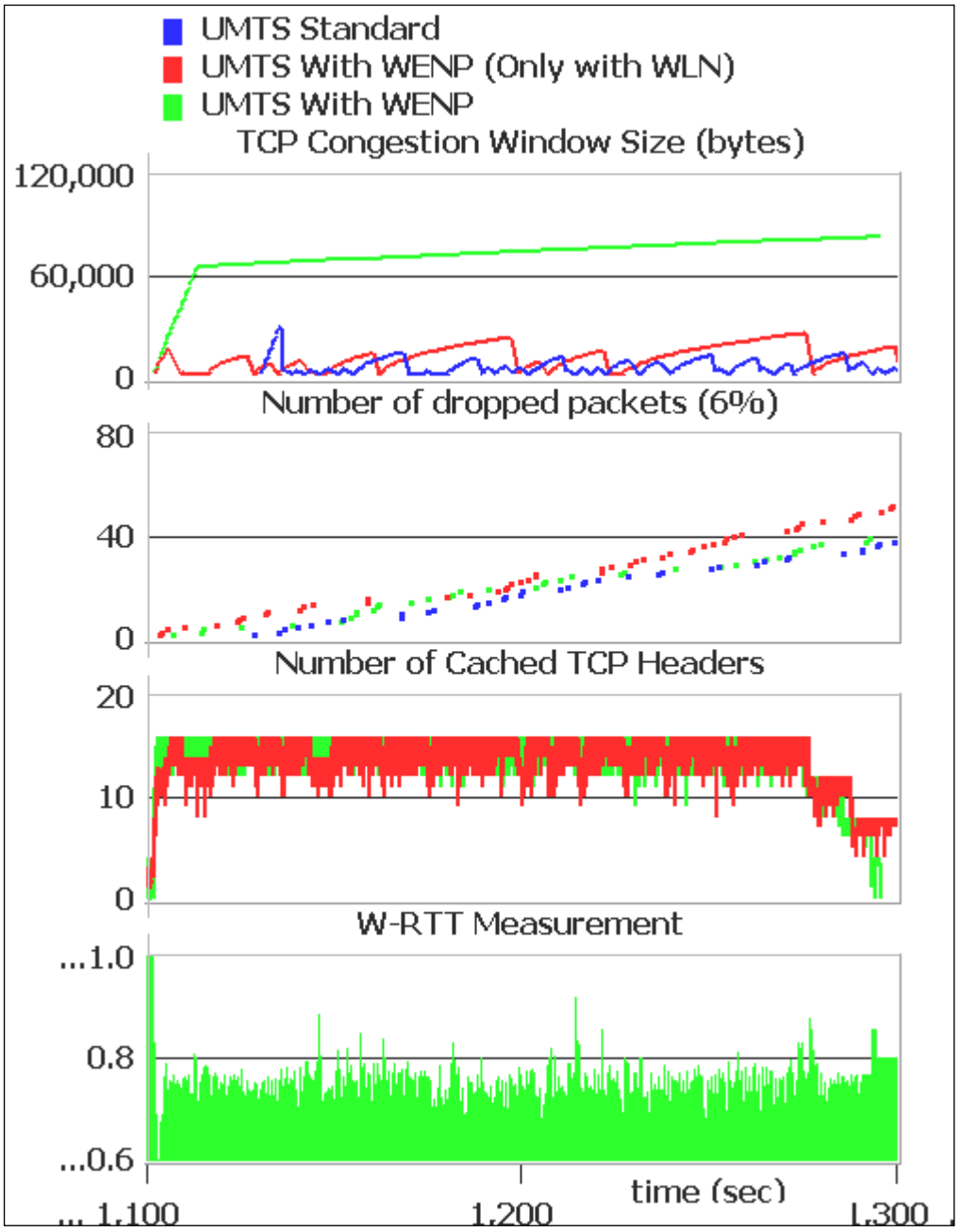

Figure 6-15 W-RTT, cached and dropped packets and TCP CWND

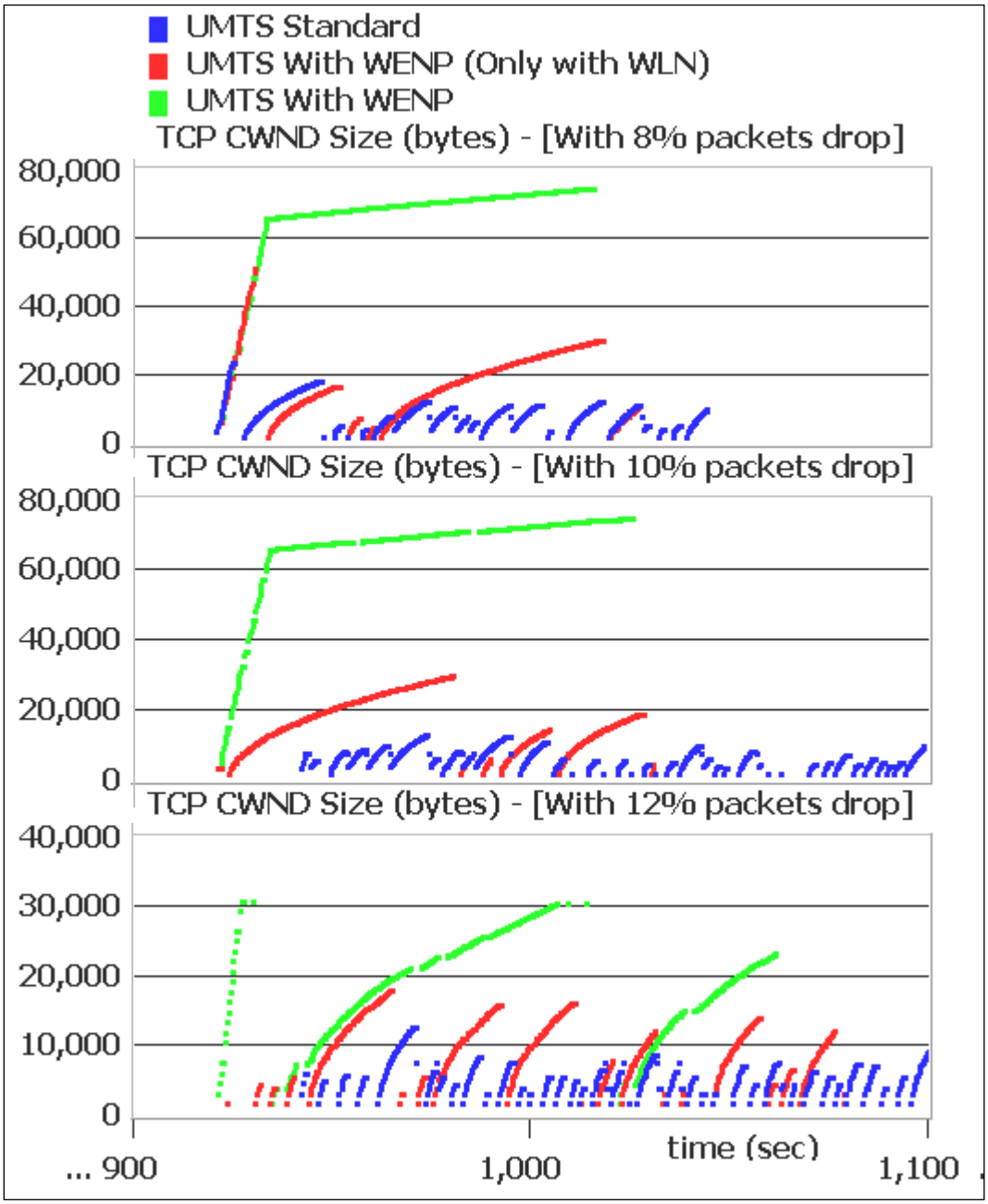

Figure 6-16 TCP CWND size responses

The performance of the proposed scheme with and without SACK option enabled is also implemented in this UMTS model and it was found that the SACK option enabled does not have any impact on the proposed scheme. It can be seen from Figure 6.17, which shows comparisons of TCP sent segment sequence number responses for different packet drops rate.

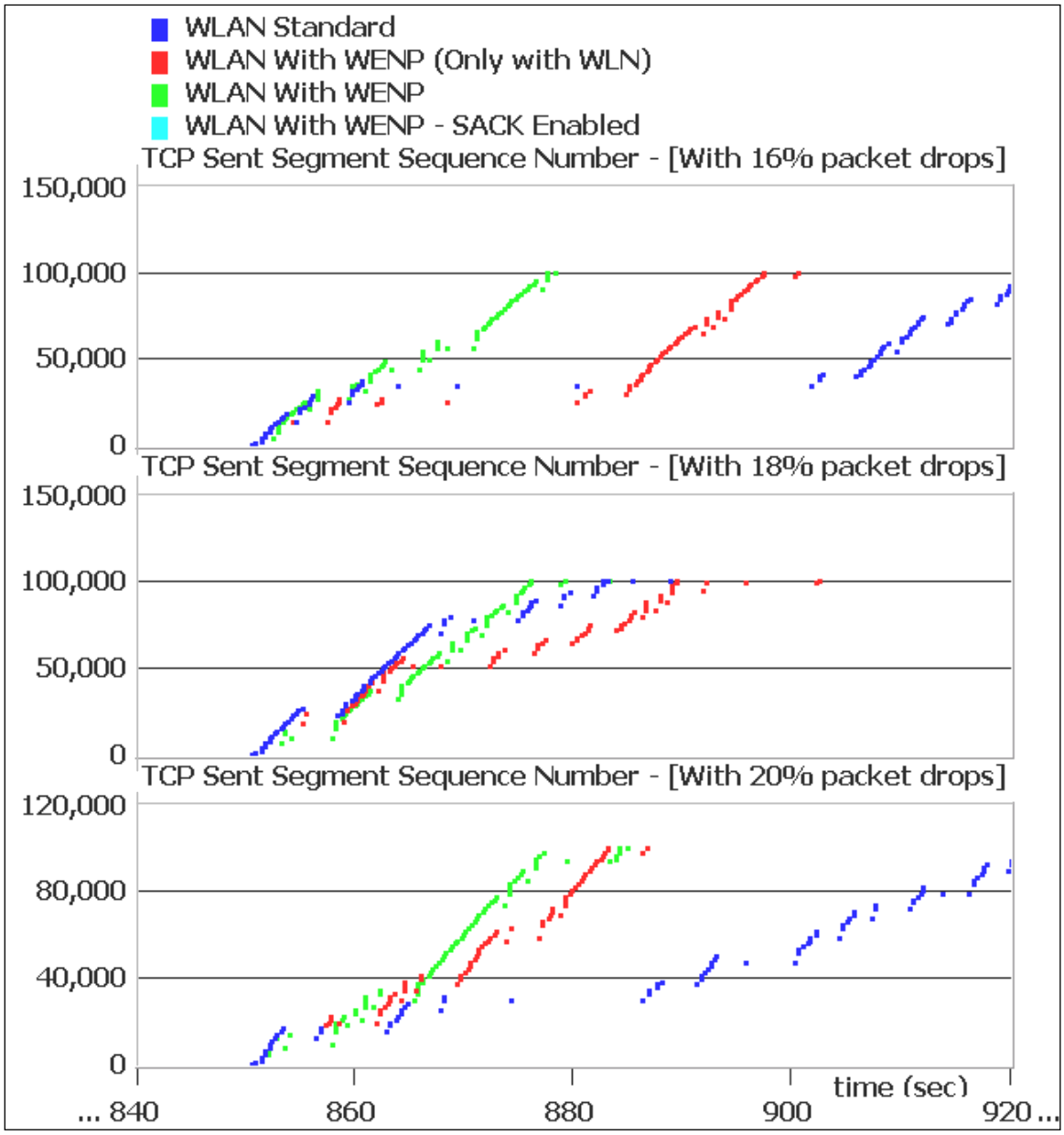

Figure 6-17 Comparisons of TCP sent segment sequence number responses

From the TCP performance summary, shown in Table 6.7, it is observed that the proposed scheme significantly improves the TCP throughput with high packet error rates. However, the rate of improvement is not proportional to the packet drops rate since wireless error recovery mechanism not only depends on the proposed scheme but also on the TCP Reno behavior; the TCP Reno cannot recover form multiple losses within a window of data. The average UMTS Node-B throughput shown in Figure 6.18 indicates that the proposed scheme has utilized the available wireless network resources efficiently.

Packet Drop Rates (%)	0	2	4	6	8	10	12	14	16	18	20
UMTS Standard TCP Throughput (Kbps)	45.8	43.2	39.5	32.4	32.0	25.3	21.7	18.3	17.6	13.6	11.2
UMTS With WLN TCP Throughput (Kbps)	45.8	43.4	40.1	34.1	37.6	33.1	26.7	24.8	22.7	20.3	22.2
UMTS With WENP TCP Throughput (Kbps)	45.8	43.7	42.1	40.9	42.6	36.7	33.5	32.2	32.1	30.6	29.7
TCP Improvement With WLN (%)	0	0.5	1.5	5.2	17.5	30.8	23	35.5	29.0	49.3	98.2
TCP Improvement With WENP (%)	0	1.2	6.6	26.2	33.1	45.1	54.4	76.0	82.4	125.0	165.2

Table 6-7 TCP performance summary

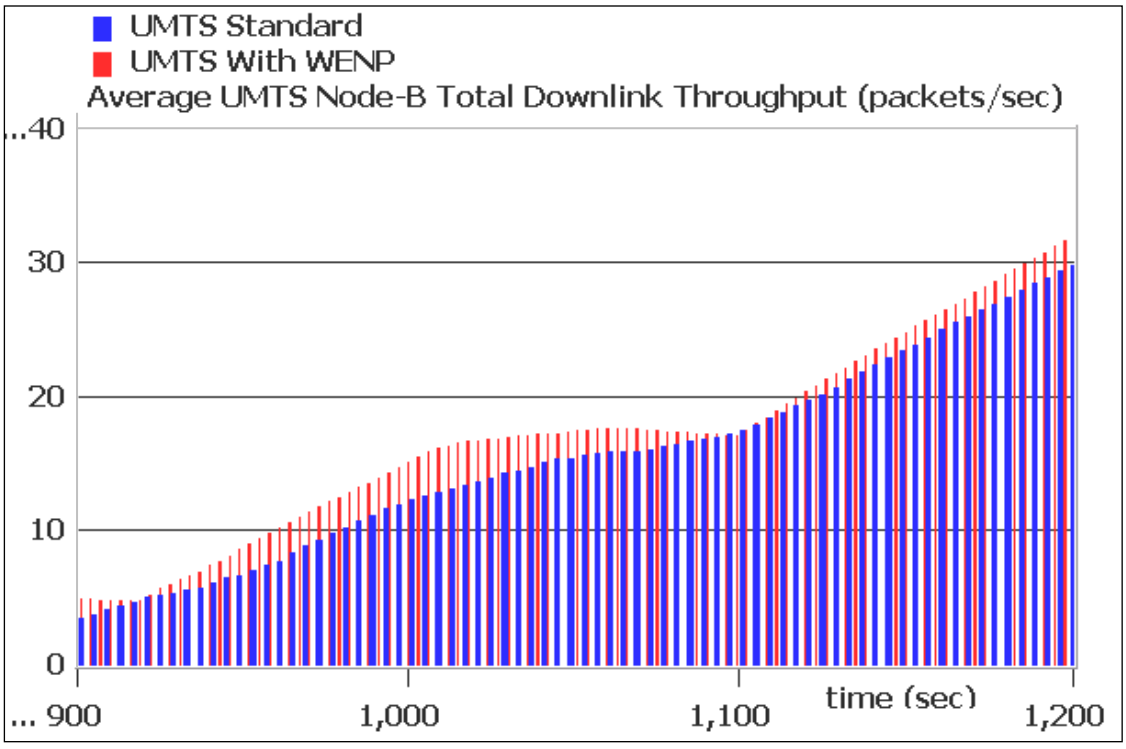

Figure 6-18 UMTS Node B throughput

6.3.2 CONCLUSIONS AND FUTURE WORK

We proposed a modified TCP Reno with a WENP mechanism and implemented it in a UMTS network modeled in OPNET. Extensive simulation studies were undertaken to compare

its performance with that of the standard TCP Reno over UMTS implementation. The simulation results showed that the new WENP scheme significantly improved the TCP performance compared to that of standard scheme. Specifically, the new scheme enabled recovery of packets lost in wireless medium, and reduced the number of spurious timeouts, by allowing the TCP sender to successfully distinguish wireless losses from congestion-related losses.

The modification to the standard TCP to early detect wireless packet loss and spurious TCP timeouts help the system to quickly recover form transmission errors and to utilize the network resources efficiently. The effect of using modified TCP Reno with the SACK option with the proposed scheme was also investigated. It was found that the proposed scheme with the TCP Reno SACK option enabled does not have any impact on the proposed scheme.

The size of the cache table required to record TCP headers is quite small and so would not add significant overhead to the RNC process. The TCP header information cached at the RNC can also be used to provide additional performance enhancement by freezing the TCP sender to handle timeouts caused by handoffs.

Chapter VII.

TCP Performance Improvement over Wireless Networks via Early Packet Loss Recovery

In this Chapter, we propose a new technique that enables the TCP to early detect packet losses, which cannot be detected and retransmitted using the standard fast retransmit mechanism, and to quickly retransmit those packets without waiting for a timeout to occur, thereby improving TCP performance. Early Packet Loss Recovery (EPLR) is achieved by considering the expected number of acknowledgements and the number of packets in a flight during the fast retransmit phase of the TCP mechanism. It adds a new flavor to the TCP fast retransmit and recovery mechanism without requiring any other modification to the standard TCP implementation.

We analyze the standard TCP implementations and outline their inability to recover from multiple packet losses within a small window of data, which is highly likely in wireless environments in Section 7.1. The proposed EPLR scheme, which modifies the TCP Reno to handle multiple losses within a window of data, and its implementation details are given in Sections 7.2 and 7.3 respectively. In Section 7.4, the proposed scheme is implemented over a UMTS network and extensive simulation studies are carried out to compare its performance with that of both TCP Reno and TCP New Reno. Based on the simulation results and the analysis, we draw our conclusion in Section 7.5.

7.1 THE STANDARD TCP MULTIPLE PACKET RECOVERY MECHANISM

There are many TCP flavors, such as Tahoe [Jacobson, 1988], Reno [Jacobson, January 1995], New Reno [S. Floyd & Henderson, April 1999] and SACK [Mathis .M, Mahdavi .J, Floyd .S, & Romanow .A, April 1996], which differ in how they react to packet loss. A packet loss is detected either by the arrival of three duplicate ACKs or the absence of an ACK for the packet within the retransmission timeout. All TCP implementations reset CWND after the retransmission timeout expiration to one MSS. However, they may proceed differently after duplicate ACKs are received.

The missing segment is always retransmitted immediately, but transmission of new or unacknowledged data depends on the selected TCP flavor. Consider a scenario illustrated in Figure 7.1, where packets P_N, P_{N+4} and P_{N+5} are lost while there is F number of packets in the flight.

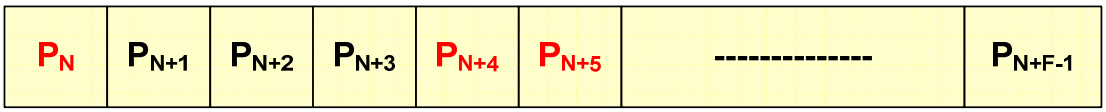

Figure 7-1 A flight of data in the network

TCP Tahoe [Jacobson, 1988] retransmits the lost packet P_N and enters into slow-start phase, setting its CWND to one MSS. The next ACK that acknowledges the packets up to P_{N+3} allows the sender to increase its CWND to two MSS and resend the packets P_{N+4} and P_{N+5}. The ACK for P_{N+4} increases the sender's CWND to three MSS and packets P_{N+6} and P_{N+7} can be sent. The ACK for P_{N+5} acknowledges packets up to P_{N+F-1}. The sender then continues transmitting new data. Notice that the packets P_{N+6} and P_{N+7} are unnecessarily retransmitted, assuming $F > 7$, and TCP Tahoe recovers from the packet losses within a window of data without retransmission timeout expiration.

TCP Reno [Jacobson, January 1995] retransmits the lost packet P_N and enters into fast recovery phase, setting its CWND to $(F/2 + 3)$ times MSS. The sender then continues receiving more duplicate ACKs and increases its CWND by one MSS for each ACK. The ACK for retransmitted packet P_N takes the sender out of fast recovery and the CWND is set to $F/2$ times MSS. The sender then waits for another three duplicate ACKs to retransmit the packet P_{N+3}. If the sender receives three duplicate ACKs, it will retransmit the packet P_{N+3} otherwise, it has to wait for the retransmission timer to expire. This increases the application response time considerably.

TCP New Reno [S. Floyd & Henderson, April 1999] retransmits the packet P_N , resets the CWND and enters into recovery phase as it does in Reno. The process then continues receiving more duplicate ACKs and increases its CWND by one MSS for each received ACK. Unlike in Reno, the ACK for the retransmitted packet P_N does not take New Reno out of the recovery process. Partial ACKs add one MSS to CWND and decrease it by the amount of acknowledged data. Notice that during the fast recovery process, it retransmits the unacknowledged data packets

whenever it receives a partial ACK without waiting for the three duplicate ACKs to arrive, which allows the process to resend the packets P_{N+4} and P_{N+5}.

TCP SACK [Mathis .M, Mahdavi .J, Floyd .S, & Romanow .A, April 1996], similar to the flavors explained above, retransmits packet PN using the fast retransmit algorithm. However, it uses a different approach to determine when and which packets are sent out during fast recovery. It calculates the amount of in-flight data based on selective acknowledgements that it has received. Data can be sent only if the amount of outstanding data is lower than the size of the CWND. Because it has information about which packets were received, it is able to resend only missing segments and then continues with transmission of unsent data.

7.1.1 SUMMARY OF TCP VARIANTS IN MULTIPLE LOSS RECOVERY

TCP Tahoe can successfully recover from multiple losses within a window of data, assuming the retransmitted segments are not lost. However, the assumption it makes that all outstanding segments are lost on receiving three duplicate ACKs leads to unnecessary retransmissions and inefficient use of valuable network resources. TCP Reno introduces a major improvement over TCP Tahoe by changing the way it reacts to detecting a packet loss through duplicate ACKs when a single packet is dropped from a window of data, but still suffers from performance problem when multiple packets are dropped from a window of data. TCP New Reno adds further improvements to the TCP Reno to be able to recover from multiple losses within a window of data. However, it can retransmit only one packet per RTT, thereby reducing the throughput performance and increasing the application response time. TCP SACK, on the other hand, can recover from multiple losses from a window of data, but requires additional bytes in the TCP header, resulting in increased overhead in bandwidth constrained wireless networks.

7.2 THE PROPOSED EPLR SCHEME

The proposed EPLR scheme takes the number of packets in a flight as the system metric to quickly fast retransmit the dropped packet that can only be recovered by TCP timeout with the existing fast retransmit and recovery algorithms, such as Reno and New Reno.

Let CWND be W, the number of dropped packets be N, where $N \geq 1$, L_I be the dropped packets within that window of data, as shown in Figure 7.2, D_I be the number of packets from L_I to P_W and the flight size be F. Number of packets can be recovered by TCP fast retransmit and recovery algorithms depend on W, N and D_I. The TCP sender can only transmit new packets if

minimum of the CWND and the receiver window is grater than F, defined as the amount of data that has been sent but not yet acknowledged. We assume that the receiver window is bigger than W for the ease of analysis.

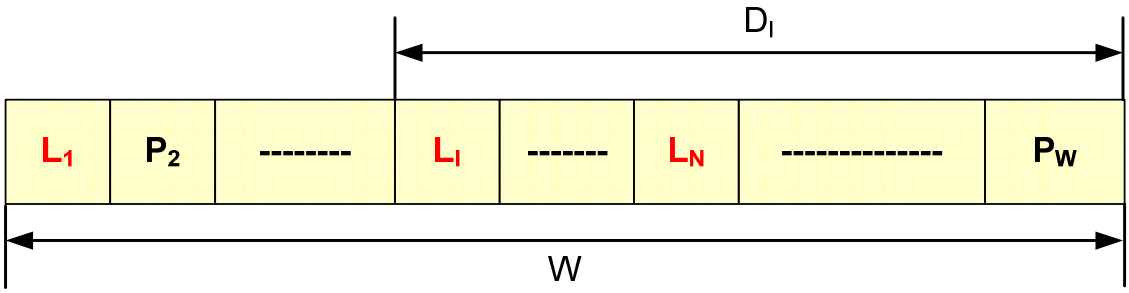

Figure 7-2 A window of data with multiple packet drops

7.2.1 TCP RENO CONGESTION WINDOW ANALYSIS

7.2.1.1 One packet recovery

In order to recover the first dropped packet, it requires that $W \geq N+3$. During this recovery process, there can be up to $W/2-N$ number of new packets transmitted, causing the flight size F to grow from W to $3W/2-N$. On receiving the ACK for the retransmitted packet L_1, F will be equal to $D_2 + (W/2-N)$ and CWND is set to $W/2$.

7.2.1.2 Two packets recovery

Notice that on receiving the first partial ACK, there can be maximum of one new packet transmitted if and only if $N-D_2 = 1$. It makes $F = CWND = W/2$. Assuming it is not the case, there will be only $W/2-N$ number of packets to be acknowledged out of $(D_2+W/2-N)$ number of packets in the network. It requires $W \geq 2(N+3)$ to be able to recover the second dropped packet. During this recovery process, $(W/4 - D_2)$ number of new packets can be transmitted, provided $W/4 > D_2$. On receiving the ACK for the retransmitted packet L_2, F will be equal to $D_3 + (W/2-N)+(W/4-D_2)$, and CWND is set to $W/4$.

7.2.1.3 Three or More packets recovery

Third packet recovery requires $W/4-D_2 \geq 3$ and allows $(W/8- D_3)$ number of new packets to be transmitted, provided $W/8 > D_3$. It can be generalized that it requires $(W/2^{N-1}-D_{N-1} \geq 3)$ to recover form the N^{th} packet, provided $(N-1)^{th}$ packet is recoverable and $N > 2$.

7.2.2 TCP NEW RENO CONGESTION WINDOW ANALYSIS

TCP New Reno only requires $W \geq N+3$ to initiate the fast retransmit and recovery algorithms. Because, once entered into the recovery phase, it can recover from multiple packet drops within a window of data by retransmitting the unacknowledged packets whenever it receives a partial ACK.

7.2.3 INTUITION BEHIND THE PROPOSED SCHEME

In order to proceed with packet transmissions, dropped packets must be retransmitted as quickly as possible. TCP Reno and New Reno fast retransmit and recovery algorithms are well defined and designed to handle this effect. However, they fail to consider situations where the fast retransmit and recovery algorithms cannot be even initiated.

- If the CWND size is less than or equal to the duplicate acknowledgement threshold, which is normally assigned to be three, either the TCP Reno or New Reno cannot even initiate the fast retransmit and leave this packet to be recovered by means of TCP timeout process.

- TCP Reno cannot initiate the fast retransmit to recover from multiple packets if $CWND < 10$.

Our insight is that if the congestion window size or the expected number of duplicate acknowledgement packets is too small to initiate the fast retransmit, it must be handled separately.

We modify the TCP Reno fast retransmit algorithm in order to recover from up to two packets within a window of data if CWND is too small to initiate the fast retransmit. Now, it comes to deciding when to retransmit the unacknowledged packet. Clearly, the TCP sender cannot confirm a received duplicate ACK was due to packet loss because packets in a flight could take different route and reach the destination out of order. Given the number of packets in a flight is F and is equal to the CWND, we can safely assume that a packet is dropped if the sender receives $(F-1)$ number of duplicate ACKs and allow the sender to quickly retransmit that packet if the flight size is too small to initiate the fast retransmit and recovery algorithms. However, due to the sender's inability to confirm the packet loss, we decide to allow the sender to transmit a new data packet by increasing the CWND by one MSS. It enables the sender to receive threshold

number of duplicate ACKs and either to fast retransmit the lost packet if a third duplicate ACK is received or to continue transmitting new data if a non duplicate ACK is received. This will considerably increase the TCP throughput and application response time while minimizing the number of TCP timeouts.

7.3 IMPLEMENTATION DETAILS OF THE PROPOSED SCHEME

TCP Reno is optimized for the case when a single packet is dropped within a window of data [Fall & Floyd, 1996] and is the most widely used TCP implementation in the Internet today [Dongkyun, Hanseok, Jeomki, & Cano, 2005]. The proposed scheme modifies the TCP Reno fast retransmit and recovery algorithm to recover form multiple losses within a window of data. Figure 7.3 shows the CWND evolution during the first packet recovery where three packets are dropped from a window of data. Note that the flight size F on receiving the first partial ACK is *(D2+W/2– 3)* even though the CWND is set to *W/2*. New packets can only be transmitted if the CWND is grater than the flight size.

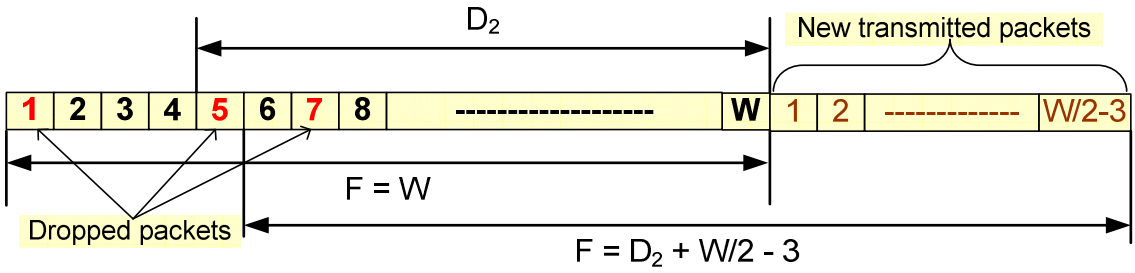

Figure 7-3 CWND evolution during the fist packet recovery

In order to recover the next dropped packet within that window, the new transmitted packets should be greater than three, which is the duplicate ACK threshold to trigger the fast retransmit and fast recovery algorithm. The proposed scheme uses the number of newly transmitted packets during the fast recovery process as the system metric and based on this, it allows the TCP sender to transmit new packets that will enable the system to receive threshold number of duplicate ACKs. This way multiple packets can be recovered without waiting for the TCP timeout to occur.

Figure 7.4 shows the flow control for EPLR ACK processing. Notice that the inflating the CWND size during the fast recovery phase plays an important role in the process of recovering multiple packets from a window of data. However, it is the successful retransmissions of the

dropped packets that take the process out of the recovery phase. If a retransmitted packet is dropped, the only way to recover form the losses is the TCP timeout process.

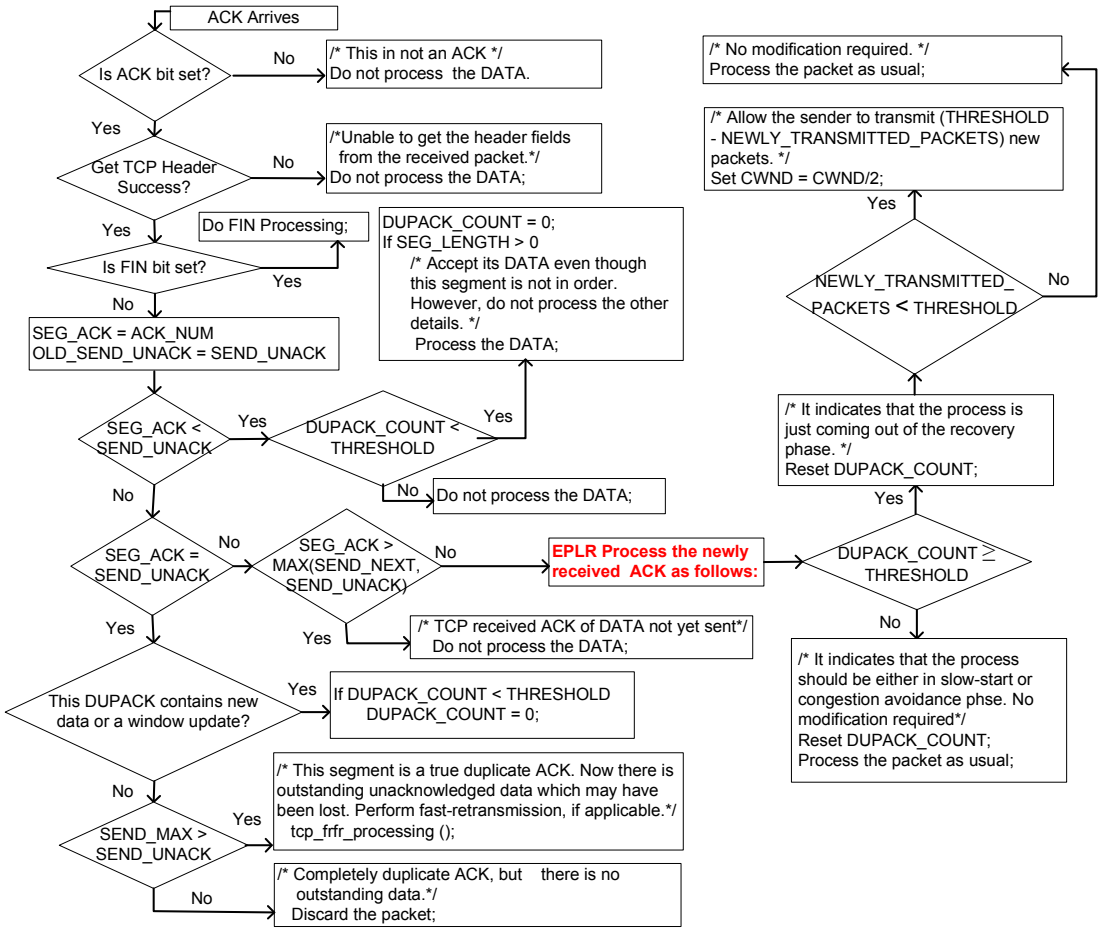

Figure 7-4 the flow control for EPLR ACK processing

7.4 PERFORMANCE EVALUATION OF THE PROPOSED SCHEME OVER UMTS NETWORK

We have implemented the proposed scheme in two UMTS models and run extensive simulations to get the mean TCP throughput with less than 5% error margin. TCP and UMTS with their default parameters are used in all simulation scenarios and an extract of the TCP and UMTS parameter values are given in Tables 7.1 and 7.2, respectively.

Maximum Segment Size (MSS)	1460 bytes
Receive Window Size at Mobile Hosts	Default
Delayed ACK Mechanism	Segment/Clock Based
Maximum ACK Delay	0.2 seconds
Maximum ACK Segments	2
Slow-Start Initial Count	1 MSS
Duplicate ACK Threshold	3
Fast Retransmit	Enabled
Fast Recovery	Reno
Selective ACK (SACK)	Disabled
Time Stamp	Disabled
Initial RTO	3.0 seconds
RTT Gain	0.125
Deviation Gain	0.25
RTT Deviation Coefficient	4.0

Table 7-1 TCP parameter values

Transmission Window Size	32
Receiver Window Size	32
SDU Discard Mode	Timer Based No Explicit
Timer MRW	140 milli seconds
Timer Discard	1500 milli seconds
Max MRW	6
Max DAT	4
In-Sequence Delivery	No
UL RLC Mode	Acknowledged Mode
DL RLC Mode	Acknowledged Mode
Admission Control Algorithm	Throughut-Based

Table 7-2 UMTS parameter values

7.4.1 SIMULATION SCENARIO 1

To demonstrate the effectiveness of the proposed scheme, the UMTS network model shown in Figure 7.5 is implemented, in turn, with different TCP fast retransmit algorithms at the standard FTP server: Reno, New Reno and Modified Reno. The FTP server is configured to generate files of 1 Mbyte size. User equipments are configured to download 1 Mbyte FTP files simultaneously with different packet drops rate as shown in Table 7.1. Data Packets coming from FTP server are dropped in UEs, at the IP layer, using a uniform probability distribution.

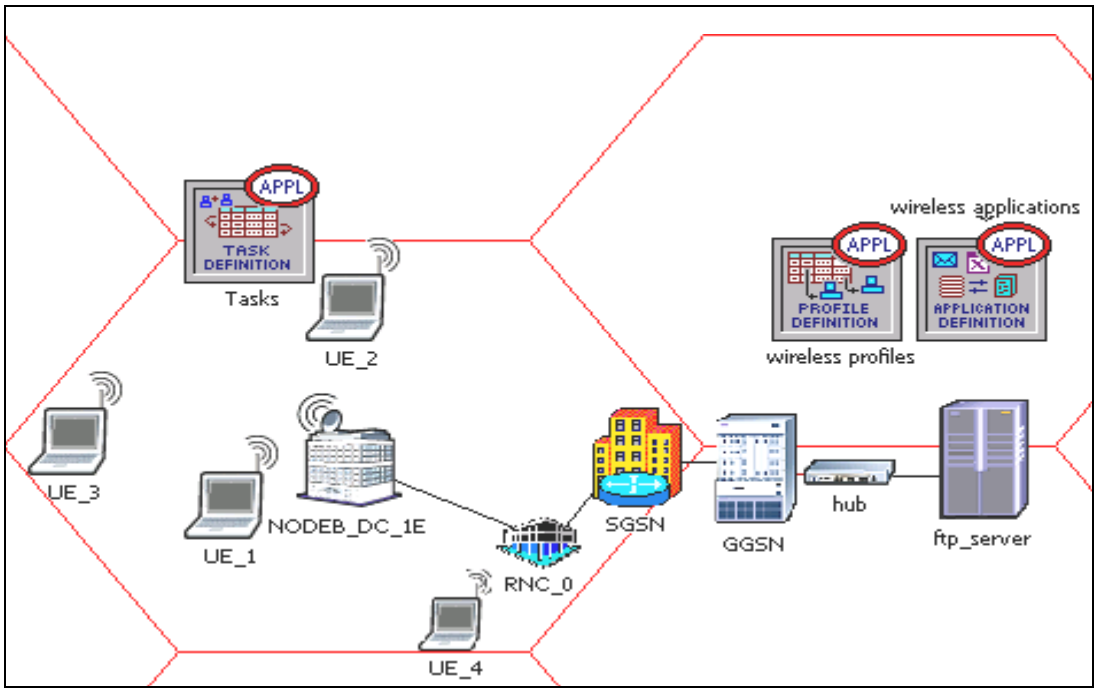

Figure 7-5 UMTS network model

User Equipments	UE-1	UE-2	UE-3	UE-4
Packet drops rate(%)	4	6	8	10

Table 7-3 User equipments configurations

7.4.1.1 Results and Observations

Figures 7.6 and 7.7, respectively, compare the TCP sent segment sequence number and the TCP CWND size responses of the proposed scheme with that of TCP Reno and New Reno implementations for different packet drops rate, espectively. A summary of the average TCP throughput and the TCP performance improvements with the proposed scheme over that of Reno and New Reno are given in Table 7.4 and Table 7.5 respectively. It can be seen that our proposed scheme improved the TCP throughput compared to that of TCP Reno significantly in all cases. This improvement can be directly attributed to the reduction of TCP timeouts, which can be observed in Figure 7.7.

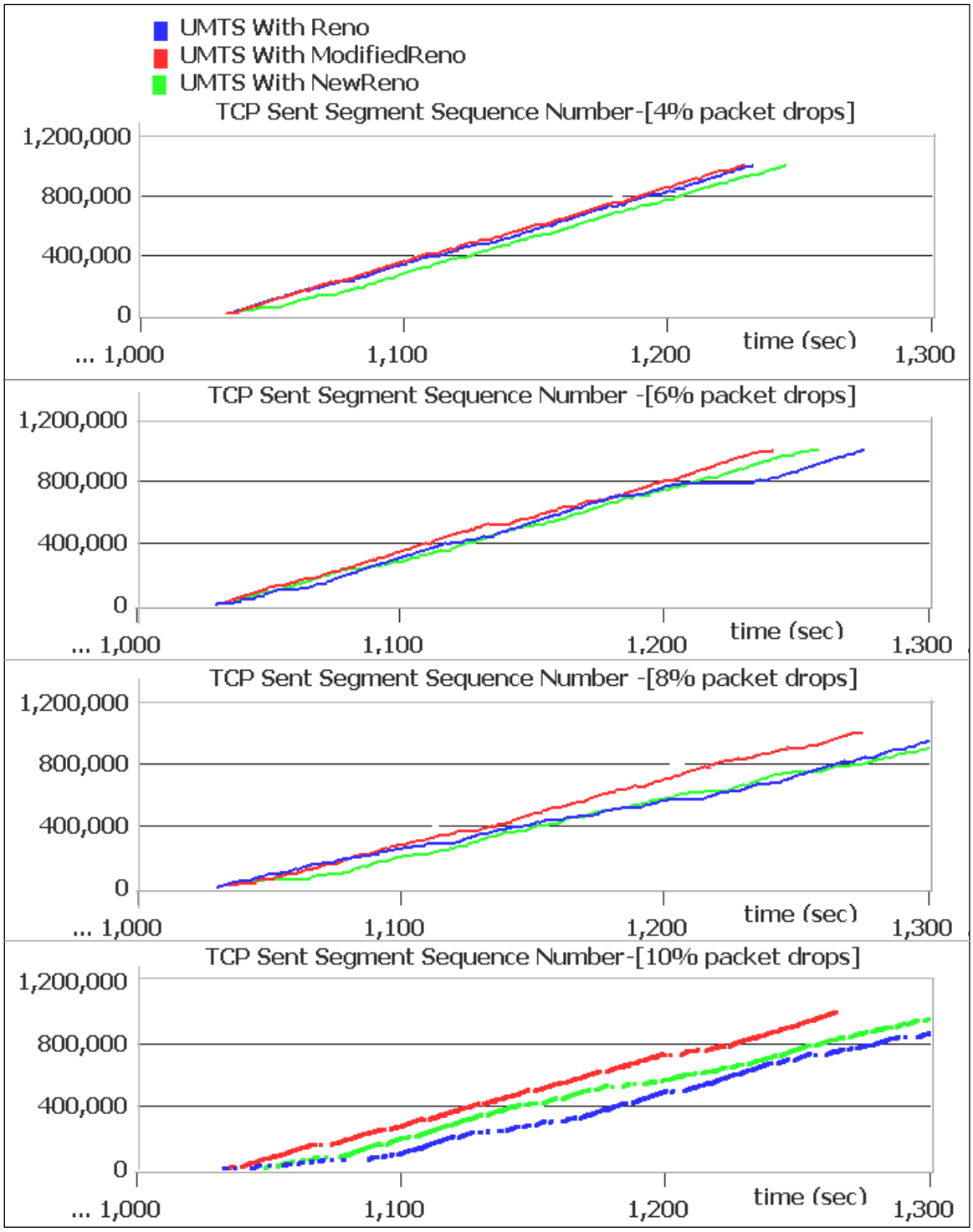

Figure 7-6 TCP sent segment sequence number

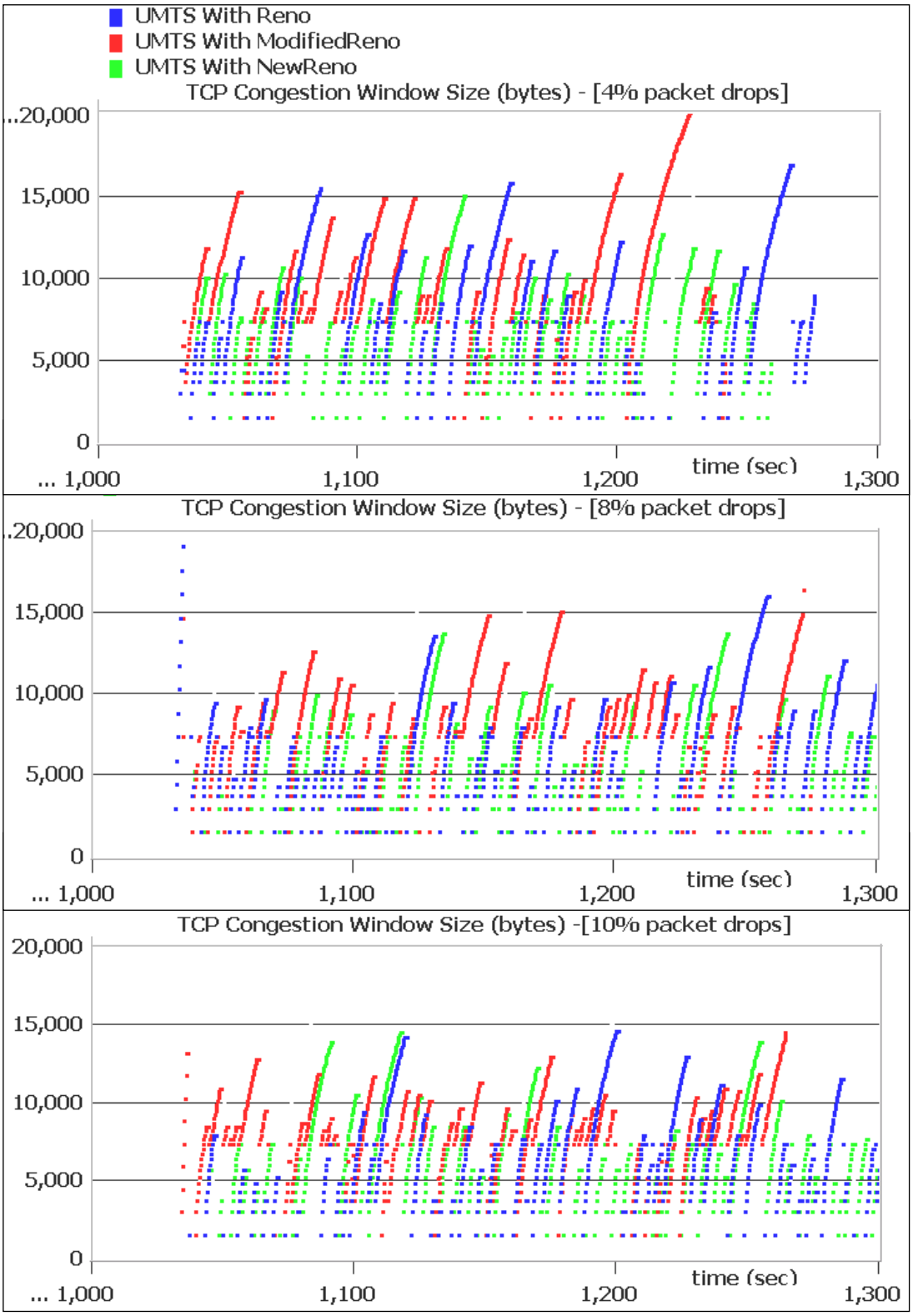

Figure 7-7 TCP CWND response

Packet drop rates (%)	4	6	8	10	Total throughput (Kbps)
TCP Reno throughput (Kbps)	37.38	32.14	29.51	28.54	127.57
TCP New Reno throughput (Kbps)	41.73	36.55	33.36	32.68	144.32
TCP Modified-Reno throughput (Kbps)	43.47	38.12	35.30	34.80	151.69

Table 7-4 Summary of average throughput performances

Packet drop rates (%)	4	6	8	10
TCP performance Improvement over Reno (%)	16.29	18.61	19.62	21.93
TCP performance Improvement over New Reno (%)	4.17	4.3	5.82	6.49

Table 7-5 Average TCP throughput performance improvement

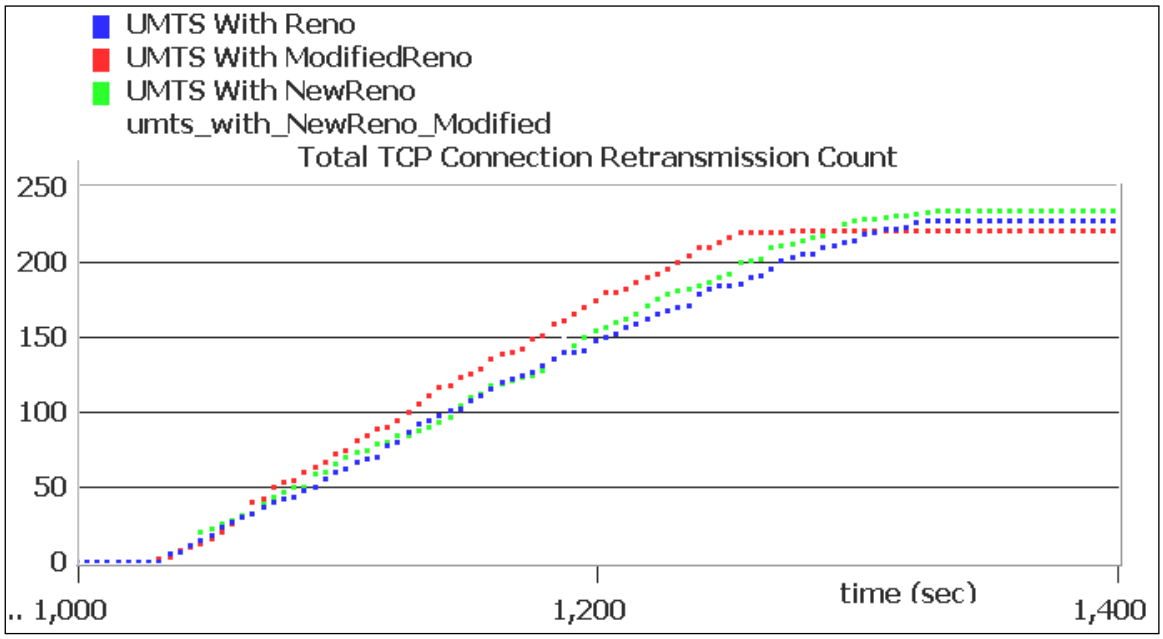

Figure 7-8 TCP retransmission count for UE-4

In order to prove the point that we have not overloaded the network, the number of retransmission counts during the file download by UE-4 is given in Figure 7.8. It can also be observed that the proposed scheme seems to start the retransmissions sooner than does TCP Reno. To explain and compare the proposed fast retransmit and recovery mechanisms with that of the standard TCP Reno, a snapshot of the TCP sent and ACK sequence number responses, during UE-4 file download, with TCP Reno and Modified Reno is given in Figure 7.9.

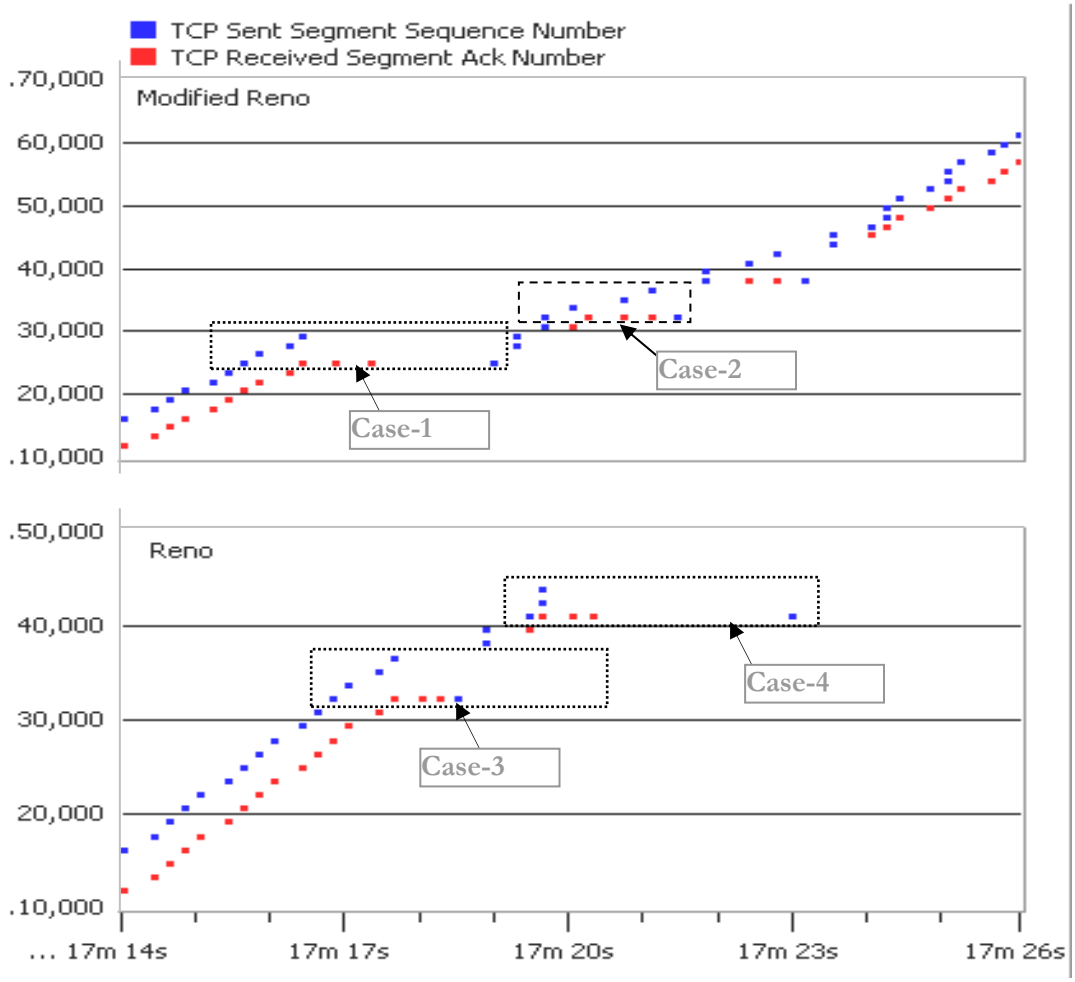

Figure 7-9 TCP sent and ACK number for UE-4

In Case-1, the sender with Modified Reno receives the first duplicate ACK with the flight size equal to four packets. It therefore does wait for the third duplicate ACK to arrive to trigger the fast retransmit. Since it has not received the third duplicate ACK in time, it times out and retransmits using the TCP timeout process.

In Case-2, the sender receives the first ACK with the flight size equal to two packets. The Modified Reno then allows the sender to transmit a new packet for each duplicate ACK. It enables the sender to receive three duplicates ACKs and to confirm the packet loss. As expected, it does receive the third duplicate ACK and the dropped packet gets retransmitted without waiting for a TCP timeout that would occur with either the Reno or New Reno retransmit mechanisms.

In Case-3, the sender with Reno receives the first duplicate ACK with the flight size is equal to four packets. It therefore does wait for the third duplicate ACK to arrive to trigger the fast retransmit. Since it has received the third duplicate ACK in time, the dropped packet gets retransmitted without a TCP timeout.

In Case-4, the sender receives the first ACK with the flight size equal to three packets. The Reno sender then waits for the third duplicate ACK to arrive to confirm the packet loss. Since the sender will never receive three duplicate ACKs, this dropped packet can only be recovered by the TCP timeout process.

Compared to TCP New Reno, our proposed scheme does always outperform in all scenarios. This is because New Reno also undergoes the same problem when the flight size is too small to trigger the fast retransmit. This can be observed from the TCP CWND response in Figure 7.7, where New Reno experiences more timeouts than the Modified Reno does. Finally, in order to show that our proposed scheme has utilized the available network resources efficiently, a comparison of the UMTS Node-B downlink throughput performance is shown in Figure 7.10.

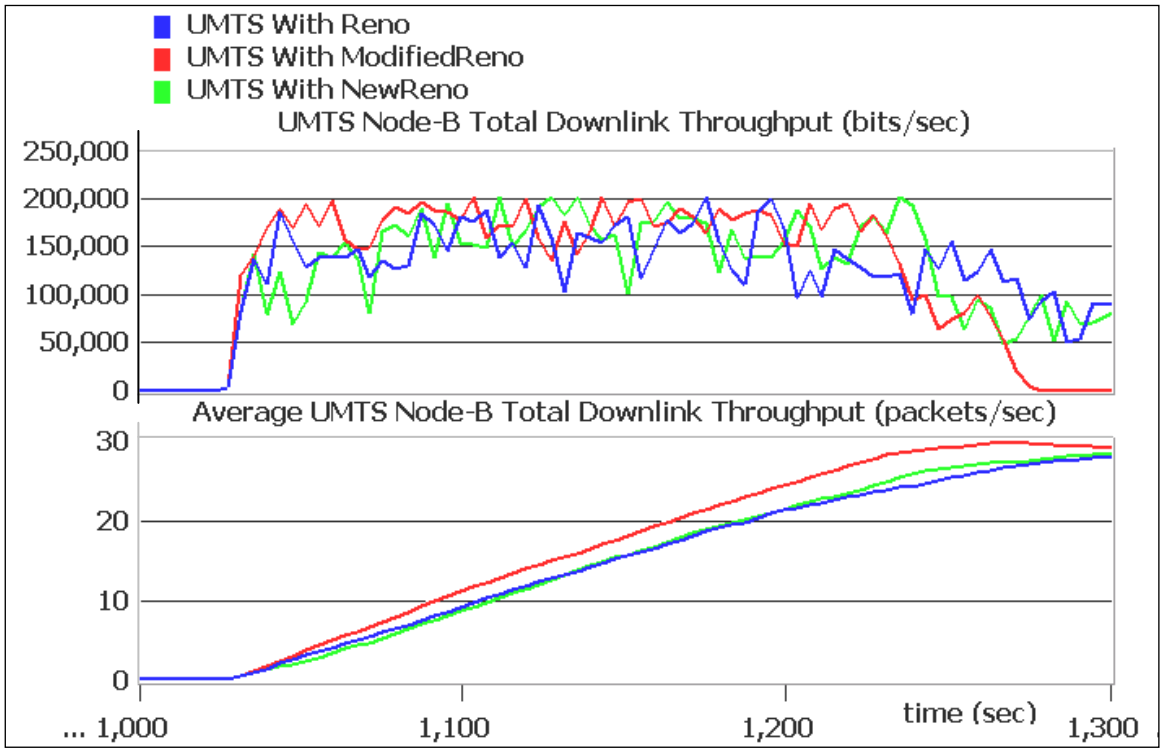

Figure 7-10 UMTS Node-B Downlink Throughput

7.4.2 SIMULATION SCENARIO 2

In this scenario, the UMTS network model shown in Figure 7.11 is implemented, in turn, with different TCP fast retransmit algorithms at the standard FTP server: Reno, New Reno and Modified Reno. The FTP server is configured to generate files of 10 Mbytes size. User equipments are configured to download 10 Mbytes FTP files simultaneously with different packet drops rate as shown in Table 7.6. Data Packets coming from FTP server are dropped in UEs, at the IP layer, using a uniform probability distribution.

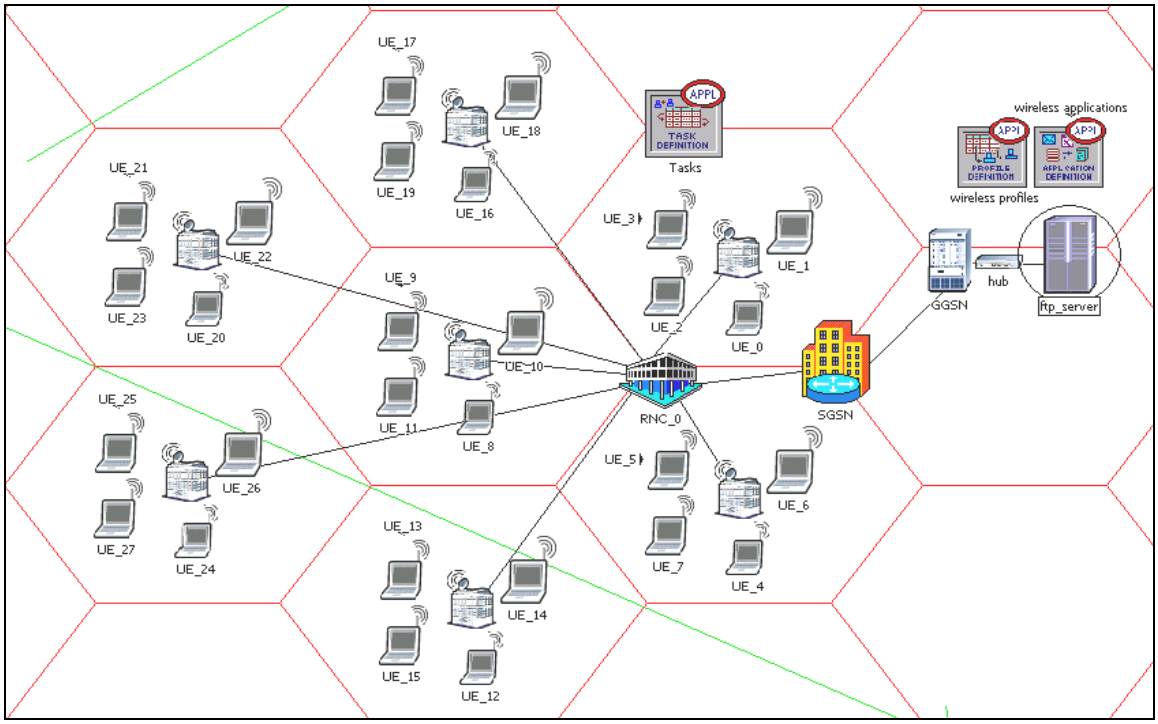

Figure 7-11 UMTS network model

User Equipments	0 - 3	4 - 7	8 - 11	12 - 15	16 - 19	20 - 23
Packet drops rate(%)	0	2	4	6	8	10

Table 7-6 UEs configurations

7.4.2.1 Results and Observations

Extensive simulations were run to get the mean TCP throughput with less than 5% error margin. Figures 7.12 shows a snapshot of the comparison of the TCP sent segment sequence number and the CWND size responses of the proposed scheme with that of TCP Reno and New Reno implementations for packet drops rate of 8% and 15%. Figure 7.13 compares the TCP sent segment sequence number responses while Figure 7.14 shows the average TCP throughput and the TCP performance improvements of the proposed scheme with that of TCP Reno and New Reno for different packet drops rate. A summary of the average TCP throughput performance with different packet drops rate is given in Table 7.7.

From Figures 7.12, 7.13 and 7.15, and Table 7.7, it can be seen that the proposed scheme improved the TCP throughput compared to that of TCP Reno and New Reno significantly. This improvement can be directly attributed to the reduction of TCP timeouts, observed in Figure 7.12. The higher the packet drops rate means higher the TCP timeouts. From Figure 7.14, it can be observed that the proposed scheme increases the TCP throughput improvement with packet drops rate.

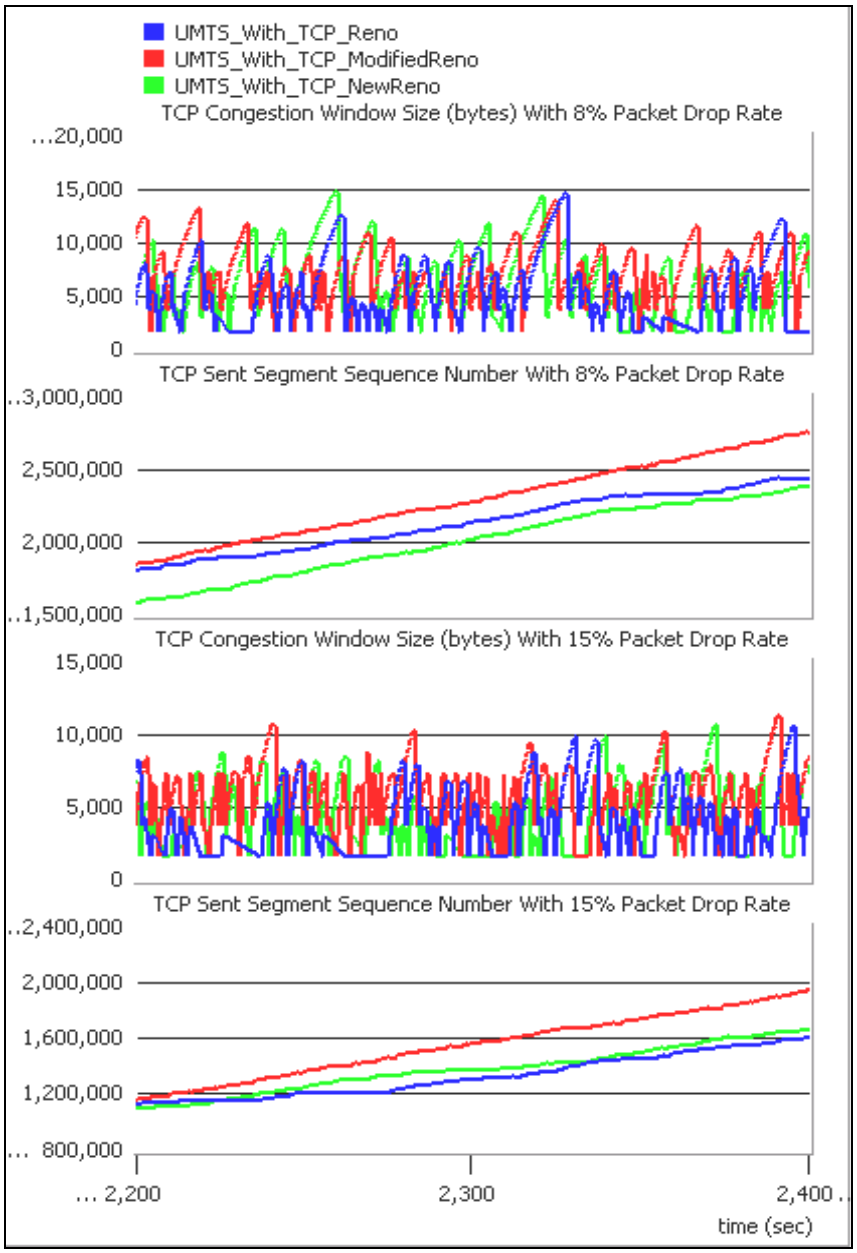

Figure 7-12 TCP sent segment sequence number

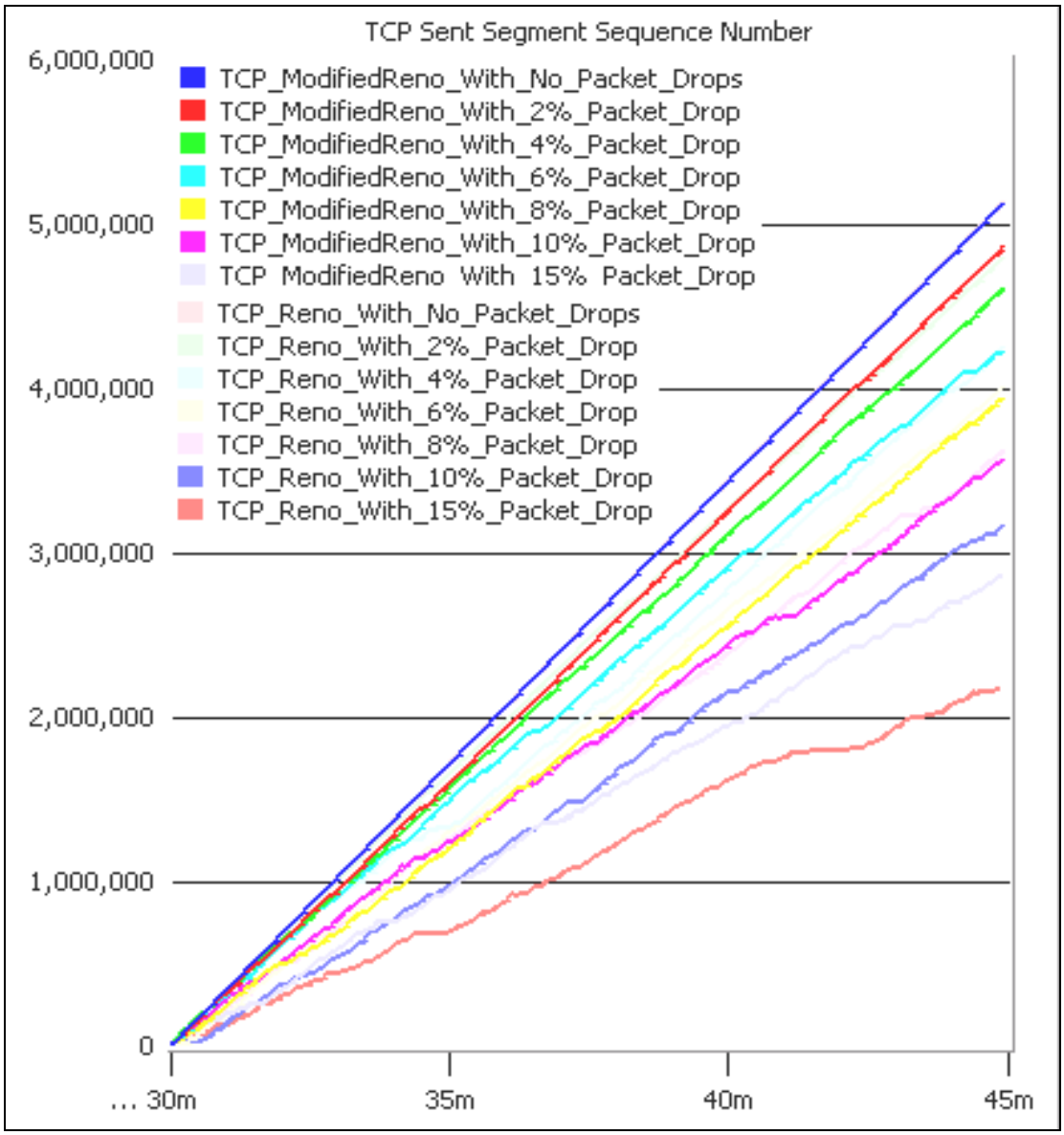

Figure 7-13 TCP sent segment sequence number responses

TCP Thoughput in (Kbps)	Packet drop rates (%)						
	0	2	4	6	8	10	15
Reno	45.79	42.93	39.30	35.55	32.23	28.44	20.92
New Reno	45.78	43.09	39.33	35.62	31.99	28.03	22.38
Modified Reno	45.79	43.41	40.60	38.03	35.13	32.11	26.54

Table 7-7 summary of the average TCP throughput performance

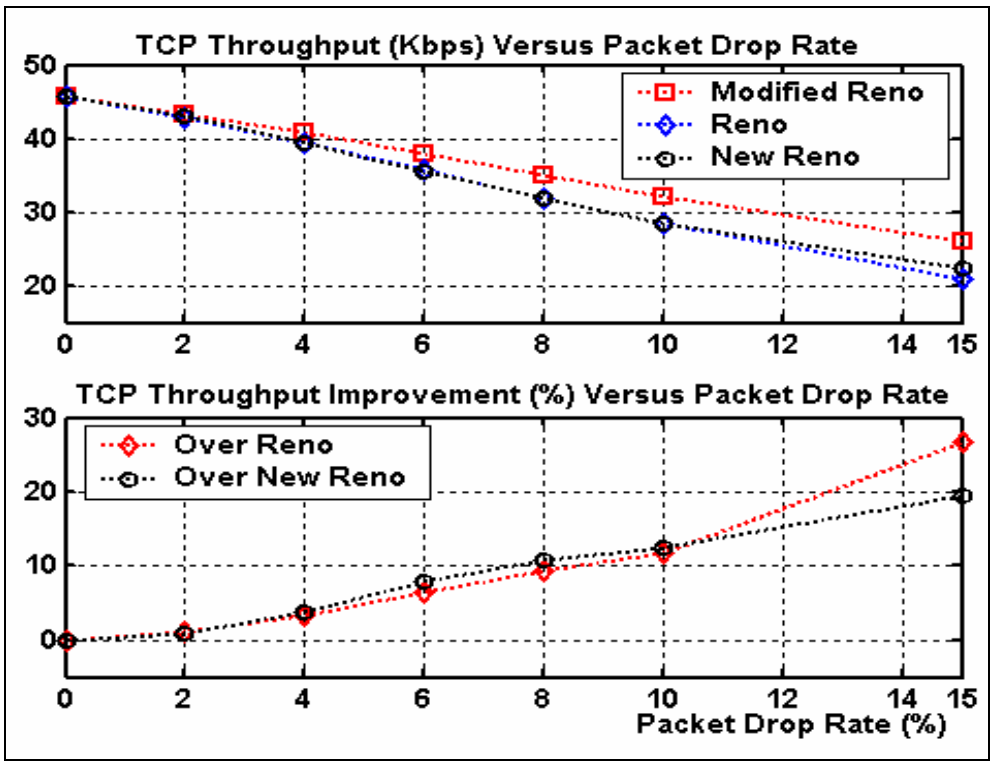

Figure 7-14 Mean TCP throughput and TCP improvement versus packet drop rates

Finally, in order to show that the proposed scheme has utilized the available network resources efficiently, a comparison of the UMTS RNC throughput performance is shown in Figure 7.15.

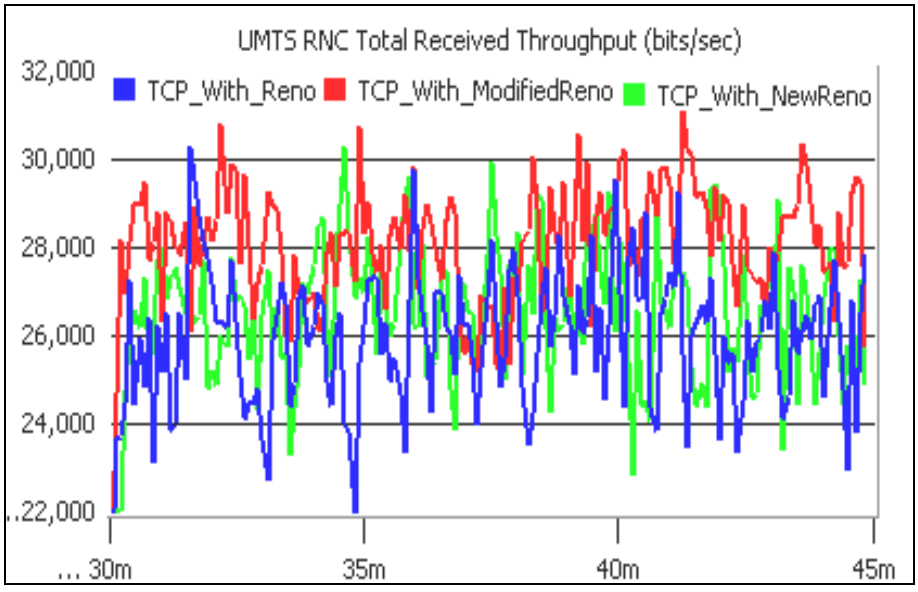

Figure 7-15 UMTS RNC Throughput

7.5 CONCLUSIONS

TCP Reno fast retransmit algorithm was modified with a new Early Timeout Detection mechanism to speed up the packet recovery process and to reduce the number of TCP timeouts over networks with heavy packet losses, such as wireless networks. Modified Reno was implemented in a UMTS network and its performance was compared with that of Reno and New Reno. Simulation results showed that Modified Reno improved the TCP performance and application response time significantly compared to that of both Reno and New Reno by reducing the TCP timeouts, which is the main cause of degradation of the TCP performance in a wireless environment.

Chapter VIII.

Analytical Model of TCP with Enhanced Recovery Mechanism for Wireless Environments

In this Chapter, we extend the analytical model for the TCP steady state throughput as a function of the network utilization factor, round trip time and packet drop rate for unlimited data transfer to capture the fast retransmit and recovery mechanisms. These mechanisms are frequently activated on connections over wireless links. We model TCP with a modified fast retransmit and recovery algorithm that allows packet recovery with smaller congestion window sizes than possible with TCP Reno or New Reno, thereby reducing the likelihood of timeouts. We also propose a further modification that dynamically adjusts its congestion window by considering the packet drop rate as the input parameter. This will further enhance the TCP performance over wireless networks and can be used to provide quality of services.

We first give an overview and comparison of analytical TCP models in Section 8.1, and then extend the analysis done in Chapter 7 to further enhance the TCP Reno fast retransmit and fast recovery algorithm in Section 8.2. Motivated by the analysis, we propose an analytical model for the TCP steady state throughput that can predict TCP performance accurately over a wide range of packet loss rates in Section 8.3. A further enhancement is proposed in Section 8.4 to dynamically adjust the TCP CWND based on the packet loss rate. In Section 8.5, we evaluate the proposed model using simulation studies in a UMTS network and demonstrate that the proposed model can predict TCP performance accurately over a wide range of packet loss rates. The conclusions drawn and directions for future work are outlined in Section 8.6.

8.1 AN OVERVIEW AND COMPARISON OF ANALYTICAL TCP MODELS

Today, most popular Internet applications, including the World Wide Web (WWW), e-mail, file transfer protocol (FTP) and remote login, use TCP as the transport protocol [Cardwell, Savage, & Anderson, 2000]. As a consequent, modeling TCP performance has attracted research attention over the past decade. Several analytical models have recently been proposed in [Padhye,

Firoiu, Towsley, & Kurose, 2000], [E. Altman, K. Avrachenkov, & C. Barakat], [Kassa & Wittevrongel, 2006], [Cardwell, Savage, & Anderson, 2000], [Mellia & Zhang, 2002] and [Mathis, Semke, & Mahdavi].

TCP models can be classified based on transfer length: short [Mellia & Zhang, 2002], long [Jitendra Padhye, Victor Firoiu y, Don Towsley, & Jim Kurose; Mathis, Semke, & Mahdavi] and arbitrary [Cardwell, Savage, & Anderson, 2000]. The transfer length determines the congestion control algorithms and the packet loss detection mechanisms that need to be incorporated in to the model. In case of short-lived transfers, TCP performance is strongly affected by the connection establishment and slow-start phases, with packet losses mostly being detected by TO. Models for long-lived transfers capture the steady state performance of TCP, which is dominated by the congestion avoidance (CA) phase, and packet loss recovery by three duplicate acknowledgements (TD) as well as timeout (TO).

In [Mathis, Semke, & Mahdavi], steady-state throughput (T) is predicted as a function of MSS, RTT and packet loss rate (p) for bulk transfer TCP flow. It only considers the congestion avoidance phase and packet loss recovery using TD and assumes that segment loss process is periodic with a constant probability of p. It implies that every segment loss is followed by the successful delivery of 1/p segments. Consequently, the evolution of CWND will follow a periodic saw-tooth pattern during the equilibrium. Given a maximum CWND of W_M the minimum value of CWND is $W_M/2$. Hence, the duration of each period is $(W_M/2 \times RTT)$ and the throughput T is gives as:

$$T = \frac{MSS}{RTT}\frac{K}{\sqrt{p}} \qquad \text{Equation 8.1}$$

Where, K is a constant that depends on the TCP receiver acknowledgement strategy. If the receiver does not employ the delayed acknowledgement, then K = √3/2.

In [Padhye, Firoiu, Towsley, & Kurose, 2000], steady-state throughput is predicted for bulk transfer TCP flow by considering packet loss detection by TD during CA phase and timeout (TO). The behavior of TCP congestion control is modeled in terms of rounds. A round starts with the transmission of the first segment within a window of data and ends when the ACK for that packet is received. Therefore, the duration of the round will be equal to one RTT period. It

assumes that the packet losses are correlated; if a segment is lost, all the remaining segments in the same round are considered to be lost. Stochastic system techniques is used to determine the expected values of the number of segments transmitted in a round and the duration of the round in terms of the loss probability p 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. The throughput T is approximated as:

$$B(p) \approx \frac{1}{RTT\sqrt{\frac{2b}{3}} + T_0 \min\left(1, 3\sqrt{\frac{2bp}{3}}\right) p\left(1 + 32p^2\right)} \qquad \text{Equation 8.2}$$

Where T_0 is the initial value of RTO and b is the number of packets that are acknowledged by the TCP receiver.

[Mellia & Zhang, 2002] propose a recursive, analytical model for the TCP short-lived flows to estimate the completion time as a function of the average loss rate and the RTT along the flow path. The connection establishment latency $E[L_{CE}]$ is calculated using (8.3) with $p_r = p_f = p_s$, where, p_f is the segment loss rate in the forward path from the server to the client, p_r is the loss rate in the reverse path, p_s is the loss rate of SYN segment and RTO_0 is the initial RTO value.

$$E[L_{CE}] = RTT + RTO_0\left(\frac{1-p_r}{1-2p_r} + \frac{1-p_f}{1-2p_f} - 2\right) \qquad \text{Equation 8.3}$$

The latency L_M^W is defined as the average time spent to successfully transmit *M* segments with an initial CWND of *W*. Note that $C_n^1 = C_1^1 + C_{n-1}^2$ since after the TCP sender receives the ACK for the first transmitted segment, it transmits the remaining segments using an initial CWND of two. The latency L_M^W is calculated recursively as a function of p, RTT and RTO. For example;

$$L_1^1 = RTT + RTO\frac{P}{1-2P} \qquad \text{Equation 8.4}$$

$$L_2^2 = RTTq^2 + qp\left(RTO + RTT + L_1^1\right) + pq\left(RTO + L_1^1\right) + p^2\left(RTO + L_2^1\right) \qquad \text{Equation 8.5}$$

Where, p is uniformly distributed and *(q = 1 – p)*

8.2 THE TCP RENO ANALYSIS

Let the CWND be W, P_W be the last packet within that window of data, the number of dropped packets be N, where N ≥ 1, and L_I, I = 1, 2, …, N index the dropped packets within that window of data, as shown in Figure 8.1. Also let D_I be the number of packets from L_I to P_W, inclusive, and the flight size, defined as the amount of data that has been sent but not yet acknowledged, be F.

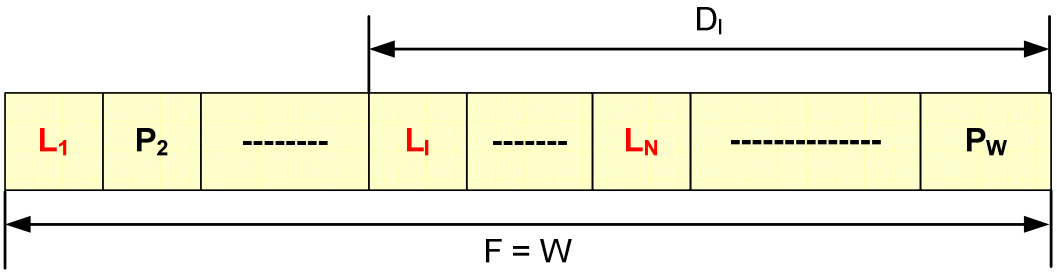

Figure 8-1 A window of data with multiple packet drops

The number of packets that can be recovered by TCP fast retransmit and recovery algorithms depend on W, N and D_I. The TCP sender can only transmit new packets if minimum of the congestion window and the receiver window is grater than F. We assume that the receiver window is larger than W for the ease of analysis.

8.2.1 FIRST PACKET RECOVERY

Recovery of the first dropped packet requires $W \geq (N+3)$. During this recovery process, there can be up to $(W/2-N)$ new packets transmitted, causing the flight size F to grow from W to $(3W/2-N)$. On receiving the ACK for the retransmission of packet L_1, F will be equal to $(D_2+W/2-N)$ and CWND is set to $W/2$.

8.2.2 SECOND PACKET RECOVERY

Notice that on receiving the first partial ACK, there can be maximum of one new packet transmitted if and only if $(N-D_2) = 1$. It makes $F = CWND = W/2$. Assuming it is not the case, there will be $(W/2-N)$ packets remaining to be acknowledged out of $(D_2+W/2-N)$ packets in the network. It requires $W \geq 2(N+3)$ to be able to recover the second dropped packet. During this recovery process, $(F/2+W/2-N-F) = (W/2-N-D_2)/2$ new packets can be transmitted, provided W

$> 2(N+D_2)$. On receiving the ACK for the retransmission of packet L_2, F will be equal to $(D_3+(W/2-N) + 1/2(W/2–N-D_2))$ and CWND is then set to $½ (D_2+W/2-N)$.

8.2.3 THIRD PACKET RECOVERY

Third packet recovery requires $½ (W/2-N-D_2) \geq 3$. Since $(F/2+½(W/2-N-D_2))–F = -¼ (W/2-N+2D_3+D_2) < 0$, there will not be any new packets transmitted during this recovery process. It implies that standard TCP Reno cannot recover from four (or more) packet drops within a window of data.

A summary of TCP Reno packet recovery mechanism is given in Table 8.1. Notice that third packet recovery depends on error pattern.

Packet Recovery	1st	2nd	3rd
Expected number of duplicate ACKs	W - N	W/2 – N	½(W/2 - N - D_2)
Requirement for packet recovery	W > N + 3	W > 2(N + 3)	W > 2(N + 6 + D_2)
Flight size before retransmission	W	D_2 + (W/2 - N)	D_3 + (W/2 - N) + ½(W/2 – N - D_2)
Congestion window after recovery	½W	½(D_2 + (W/2 - N))	½(D_3 + (W/2 - N) + ½(W/2 – N - D_2))
Flight size after recovery	D_2 + (W/2 - N)	D_3 + (W/2 - N) + ½(W/2 – N - D_2)	D_4 + (W/2 - N) + ½(W/2 – N - D_2)

Table 8-1 Summary of TCP Reno congestion window analysis

8.2.4 REQUIRED MODIFICATIONS

In order to proceed with packet transmissions, dropped packets must be retransmitted as quickly as possible. TCP Reno and New Reno fast retransmit and recovery algorithms are well defined and designed to handle this effect. However, they fail to consider situations where the fast retransmit and recovery algorithms cannot be even initiated.

- If the CWND size is less than or equal to the duplicate ACK threshold, which is normally assigned to be three, either the TCP Reno or New Reno cannot even initiate the fast retransmit and leave this packet to be recovered by means of TCP timeout process.

- TCP Reno cannot initiate the fast retransmit to recover from multiple packets if CWND < 2(N+3).

Our insight is that if the CWND size is too small to initiate the fast retransmit, it must be handled separately.

We modify the TCP Reno fast retransmit algorithm in order to recover from packet drops within a window of data, which can only be recovered by a TCP timeout process in the TCP Reno implementation. We assume that the retransmitted packets are not dropped. If the retransmitted packets are dropped, these packets can only be recovered using TCP timeout process.

8.2.4.1 Recovery of a single packet drop if CWND < 4

The TCP sender cannot confirm a received duplicate ACK was due to packet loss because packets in a flight could take different route and reach the destination out of order. Given the number of packets in a flight is F and is equal to the CWND, we can safely assume that a packet is dropped if the sender receives *(F–1)* number of duplicate ACKs and allow the sender to quickly retransmit that packet if the flight size is too small to initiate the fast retransmit and recovery algorithms. However, due to the sender's inability to confirm the packet loss, we decide to allow the sender to transmit new data packets by increasing the CWND by one MSS. It enables the sender to receive threshold number of duplicate ACKs and either to fast retransmit the lost packet if a third duplicate ACK is received or to continue transmitting new data if a non duplicate ACK is received. This will considerably increase the TCP throughput and application response time while minimizing the number of TCP timeouts.

8.2.4.2 Recovery of multiple packet drops if CWND > (N+3)

On receiving a partial ACK, the TCP sender can confirm the next packet drop within that window of data and can retransmit that packet without waiting for the third duplicate ACK to arrive. From the TCP Reno congestion window analysis, it can be observed that the flight size during the fast retransmit process depends on the position of the dropped packets except for the

first packet recovery; flight size during the first, second and third packet recovery is W, $(D_2+W/2-N)$ and $D_3+½(W/2-N-D_2)$ respectively. We modify the TCP Reno to handle the multiple packet drops as follows.

- On receiving the partial ACK, it retransmits the next unacknowledged packet as TCP New Reno does.

- It triggers the fast retransmit algorithm only during the first packet recovery, where slow-start threshold (ssthresh) is set to α times the flight size. We define α to be the network utilization factor, which is normally assigned the value 0.5.

- It resets the CWND to the slow-start threshold on receiving partial ACKs, as it does in TCP Reno, and when the process first gets out of recovery process.

- No change is made to the fast recovery process

8.3 PROPOSED ANALYTICAL MODEL FOR THE TCP STEADY STATE THROUGHPUT

We drive a TCP steady state throughput model, by considering long-lived TCP flow, with the assumptions that packets transmitted during the fast retransmit and recovery phase are not dropped and the CWND is not limited by the receiver's advertised flow control window. We define a round to be the period between the start of consequence congestion avoidance (CA) phases as shown in Figure 8.2. A round includes a CA phase and a fast retransmit and recovery phase. Figure 8.2 shows the CWND evolution during the i^{th} round. The system variables are defined as follows.

- Congestion window size at the end of CA phase (W_i)
- Number of packets transmitted during CA phase including the dropped packets (S_i)
- Number of packets (δS_i) transmitted during the last RTT period (δT_i)
- Number of RTTs until the start of the last RTT period during CA phase (T_i)
- Total number of RTTs in the round (P_i)
- Time from the start of the last RTT and to the first packet retransmission (δT_i)
- Total number of packets transmitted including retransmitted and dropped packets (X_i)
- Number of segments after which a dataless ACK will be sent (d)

- Number of packets dropped (N_i)
- Packet drop rate (p)
- The network utilization factor (αs)
- Number of RTTs during the recovery process (R_i)

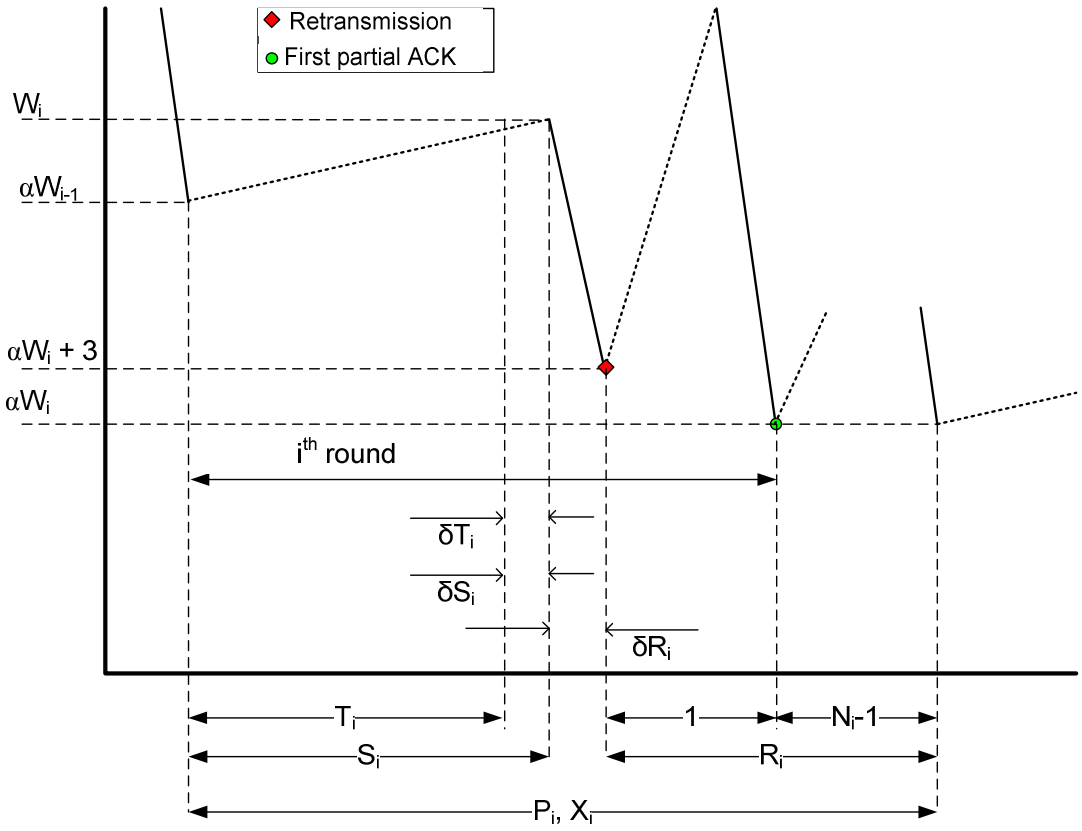

Figure 8-2 CWND evolution

We develop the model based on the CWND evolution shown in Figure 8.2. It requires N_i number of RTTs to recover from N_i number of packets within a window of data.

$$R_i = N_i \qquad \text{Equation 8.6}$$

For simplicity, we ignore the term δR_i , which is the time elapsed from the last ACK to the third duplicate ACK, and obtain the period of the i^{th} round as;

$$P_i = T_i + N_i + \delta T_i + \delta R_i \approx T_i + N_i + \delta T_i \qquad \text{Equation 8.7}$$

There will be $\alpha W_i - N_i + 1$ packets, including the first dropped packet, transmitted during the first packet recovery process. On receiving partial ACKs, it will retransmit the next dropped

packet. When the process gets out of the recovery phase and enters into the next round, there will be one new packet transmitted if $N_i=1$, otherwise αW_i new packets will be transmitted.

$$\left.\begin{aligned} X_i &= S_i + \alpha W_i + 1 \text{ if } N_i = 1 \\ X_i &= S_i + 2\alpha W_i \quad \text{if } N_i > 1 \end{aligned}\right\} \qquad \text{Equation 8.8}$$

During CA phase, CWND will grow by 1/d for each RTT period. Assuming, T_i is much bigger than δT_i , we obtain

$$W_i = \alpha W_{i-1} + \frac{T_i + \delta T_i}{d} \approx \alpha W_{i-1} + \frac{T_i}{d} \qquad \text{Equation 8.9}$$

S_i will be the sum of packets transmitted during T_i and δT_i.

$$S_i = \left(\alpha W_{i-1} \times T_i + \sum_{K=0}^{T_i/d-1} dK \right) + \delta S_i$$

$$S_i = \alpha W_{i-1} \times T_i + \frac{T_i}{2}\left(\frac{T_i}{d} - 1\right) + \delta S_i$$

Using (8.9), we obtain

$$S_i = \frac{T_i}{2}\left(W_i + \alpha W_{i-1} - 1\right) + \delta S_i \qquad \text{Equation 8.10}$$

We consider W_i to be a Markov regenerative process with rewards X_i and obtain the long-term steady-state TCP throughput B as in [Padhye, Firoiu, Towsley, & Kurose, 2000]

$$B = \frac{E[X]}{E[P]} \qquad \text{Equation 8.11}$$

Where, E[X] and E[P] are the expected value of number of packets and RTTs during the round respectively. With the assumption that T_i and W_i are mutually independent sequences of

identically distributed random variables, we obtain (8.12)–(8.15) from (8.7)–(8.10) respectively. We also consider that δS_i is uniformly distributed between 1 and W_i and have $E[\delta S] = E[W]/2$ and $E[\delta T] = RTT/2$.

$$E[P] = \left(E[T] + E[N] + \frac{1}{2} \right) RTT$$ Equation 8.12

$$\left.\begin{aligned} E[X] &= E[S] + \alpha E[W] + 1 \quad \text{if } N_i = 1 \\ E[X] &= E[S] + 2\alpha E[W] \quad \text{if } N_i > 1 \end{aligned}\right\}$$ Equation 8.13

$$E[W] = \alpha E[W] + \frac{E[T]}{d}$$

$$E[T] = d(1-\alpha)E[W]$$ Equation 8.14

$$E[S] = \frac{E[T]}{2}\left((1+\alpha)E[W] - 1\right) + \frac{E[W]}{2}$$ Equation 8.15

It can be shown that the mean number of packets successfully acknowledged before a loss occurs is 1/p [Padhye, Firoiu, Towsley, & Kurose, 2000]. It follows that total number of packets transmitted during CA phase is:

$$E[S] = \frac{1}{p} + E[W] - 1$$ Equation 8.16

From (8.14), (8.15) and (8.16), we obtain

$$E[W] = \frac{1 + d(1-\alpha)}{2d(1-\alpha^2)} + \sqrt{\left(\frac{1 + d(1-\alpha)}{2d(1-\alpha^2)} \right)^2 + \frac{2(1-p)}{pd(1-\alpha^2)}}$$ Equation 8.17

From (8.11), (8.12), (8.13), (8.14) and (8.17),

$$\left.\begin{aligned} B &= \frac{\left(\frac{1}{p}+(1+\alpha)E[W]\right)}{\left(1+\frac{1+2d(1-\alpha)}{2}E[W]\right)RTT} \quad \text{if E[N] = 1} \\ B &= \frac{\left(\frac{1-p}{p}+(1+2\alpha)E[W]\right)}{\left(E[N]+\frac{1+2d(1-\alpha)}{2}E[W]\right)RTT} \quad \text{if E[N]>1} \end{aligned}\right\} \qquad \text{Equation 8.18}$$

If d = 2, (17) yields to

$$E[W] = \frac{3-2\alpha}{4(1-\alpha^2)} + \sqrt{\left(\frac{3-2\alpha}{4(1-\alpha^2)}\right)^2 + \frac{(1-p)}{p(1-\alpha^2)}} \qquad \text{Equation 8.19}$$

Standard TCP implementation assigns α the value of 0.5. Recall that recovery of multiple packets from a window of data depends on the number of newly transmitted packets during the fast recovery phase. The network utilization factor α is the main factor that limits the number of new packets transmitted during the fast recovery phase, assuming the receiver window is bigger than the CWND size. Packet losses in network with wireless links can be bursty resulting in multiple losses within a window of data. Having larger value for α can help the system to recover from multiple losses. However, it may overload the network if the losses are due to congestion. We use the packet drop rate p to dynamically calculate the value of α as explained in the next Section.

8.4 PROPOSED SCHEME FOR DYNAMICALLY ADJUSTING THE TCP CWND

Standard TCP implementation assigns α the value of 0.5. In Equations (8.17) to (8.19), we also assign α the value of 0.5 if $p \leq 0.01$, otherwise, it is assigned dynamically. Network with wireless links has high bit-error rates, which contributes to high packet drop rates and significantly degrades the TCP performance. Figure 8.3 shows the CWND size versus packet drop rate, obtained from (8.19), with different values for α, from 0.5 to 0.85 in step of 0.05. From Figure 8.3, we can obtain α as a function of packet drop rate for desired CWND improvement. Figure 8.4 shows CWND size versus packet drop rate with α, derived as a function of packet drop rate using MATLAB curve fitting tool, to provide 10 percent CWND size improvement. It should be

noted that (8.20) is applied only for high packet drop rate and can be used to provide quality of service with different CWND improvement rates.

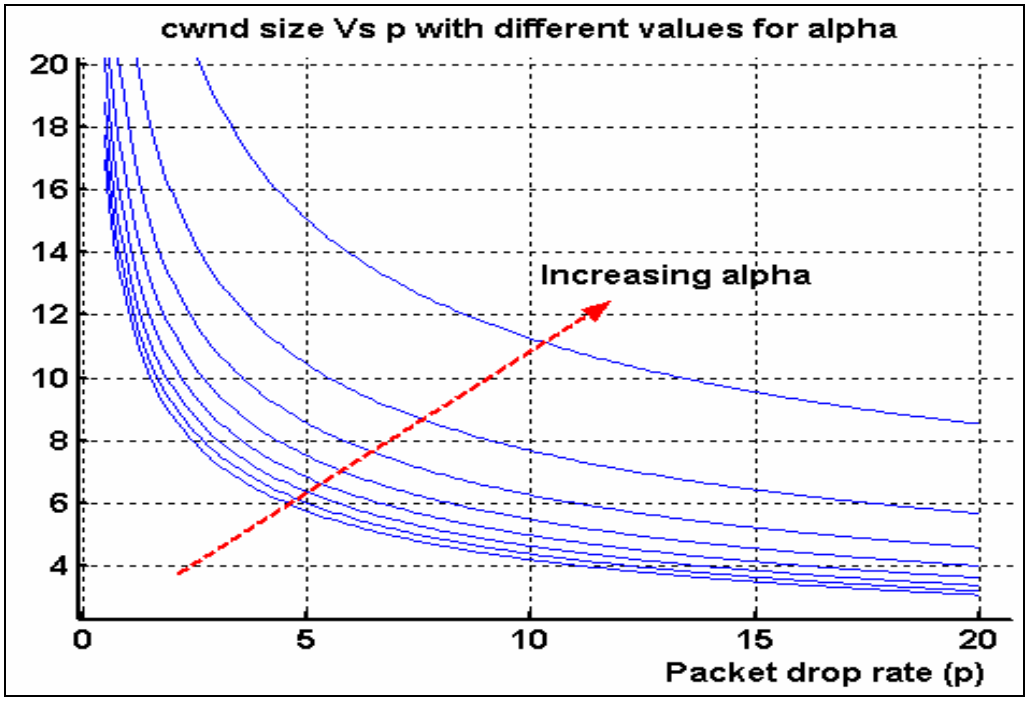

Figure 8-3 CWND versus packet drop rate (p)

$$\alpha = 0.00033p + 0.62$$ Equation 8.20

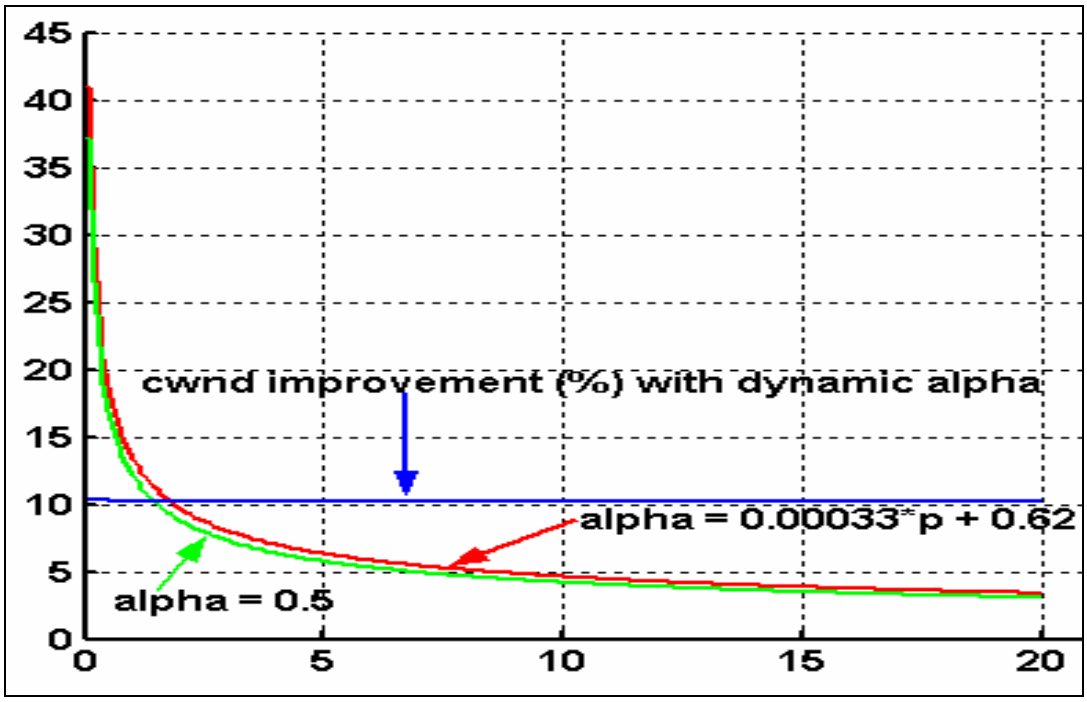

Figure 8-4 CWND size and CWND improvement versus packet drop rate (p) with dynamic network utilization factor (α)

8.5 EVALUATION OF THE PROPOSED MODEL OVER A UMTS NETWORK

The proposed scheme is implemented in OPNET by adding a new TCP flavor, called Modified Reno, to the TCP fast retransmit and recovery algorithms. We need to calculate the packet drop rate to dynamically adjust the CWND, given in (8.20). The packet drop rate is calculated with the use of a weighted average by the Average Loss Interval method explained in [Sally Floyd, Handley, Padhye, & Widmer, August 2000]. The average loss interval $\hat{s}_{(1,n)}$ is defined as a weighted average of the last n interval as follows:

$$\hat{s}(1, n) = \frac{\sum_{i=1}^{n} w_i s_i}{\sum_{i=1}^{n} w_i} \qquad \text{Equation 8.21}$$

For weights w_i:

$$sw_i = 1, \text{ if } 1 \le i \le n/2 \text{ and } w_i = 1 - \frac{i - n/2}{n/2 + 1}, \text{ if } n/2 < i \le n$$

Considering the most recent packet drop, the average loss interval ($\hat{s}$) is calculated as:

$\max(\hat{s}_{(1,n)}, \hat{s}_{(0,n-1)})$ and the packet drop rate is $1/\hat{s}$.

To demonstrate the effectiveness of the proposed scheme, the UMTS network model shown in Figure 8.5 is implemented, in turn, with TCP Reno and Modified Reno fast retransmit algorithms at the standard FTP server. The FTP server is configured to generate files of 10 Mbyte size. User equipments are configured to download FTP files simultaneously with different packet drop rates as shown in Table 8.2. Data Packets coming from FTP server are dropped in UEs, at the IP layer, using a uniform probability distribution. UMTS and TCP with their default parameters [OPNET Technologies Inc] are used in all simulation scenarios and an extract of the UMTS and TCP parameter values are given in Tables 8.3 and 8.4 respectively.

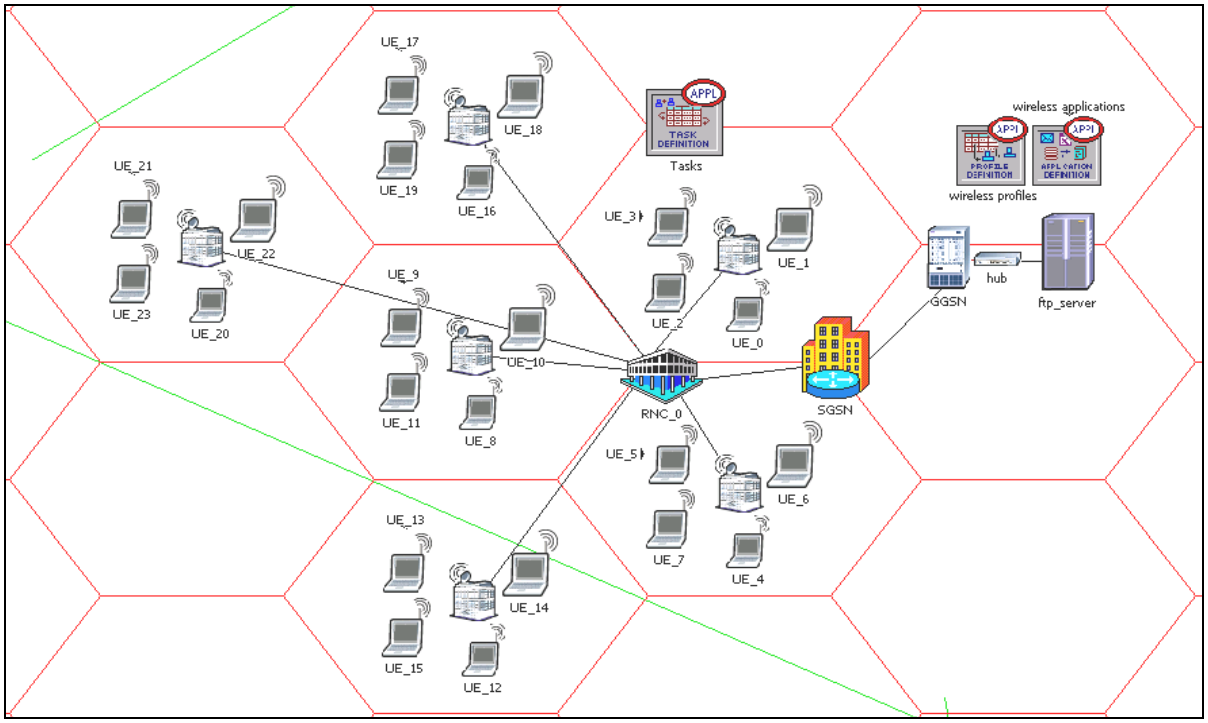

Figure 8-5 UMTS network model

User Equipments	0 - 3	4 - 7	8 - 11	12 - 15	16 - 19	20 - 23
Packet drops rates(%)	0	2	4	6	8	10

Table 8-2 UEs configurations

Transmission Window Size	32
Receiver Window Size	32
SDU Discard Mode	Timer Based No Explicit
Timer MRW	140 milli seconds
Timer Discard	1500 milli seconds
Max MRW	6
Max DAT	4
In-Sequence Delivery	No
UL RLC Mode	Acknowledged Mode
DL RLC Mode	Acknowledged Mode
Admission Control Algorithm	Throughut-Based

Table 8-3 Selection of UMTS RNC parameters

Maximum Segment Size (MSS)	1460 bytes
Receive Window Size at Mobile Hosts	Default
Delayed ACK Mechanism	Segment/Clock Based
Maximum ACK Delay	0.2 seconds
Maximum ACK Segments	2
Slow-Start Initial Count	1 MSS
Duplicate ACK Threshold	3
Fast Retransmit	Enabled
Fast Recovery	Reno
Selective ACK (SACK)	Disabled
Time Stamp	Disabled
Initial RTO	3.0 seconds
RTT Gain	0.125
Deviation Gain	0.25
RTT Deviation Coefficient	4.0

Table 8-4 Selection of TCP parameters

8.5.1 RESULTS AND OBSERVATIONS

Extensive simulations were run to get the mean TCP throughput with less than 5% error margin. Figure 8.6 shows a snapshot of the comparison of the TCP CWND size and the TCP sent segment sequence number responses of the proposed scheme with that of TCP Reno implementations for packet drop rate of 10%. It can be observed that the proposed scheme significantly reduced the number of TCP timeouts as expected.

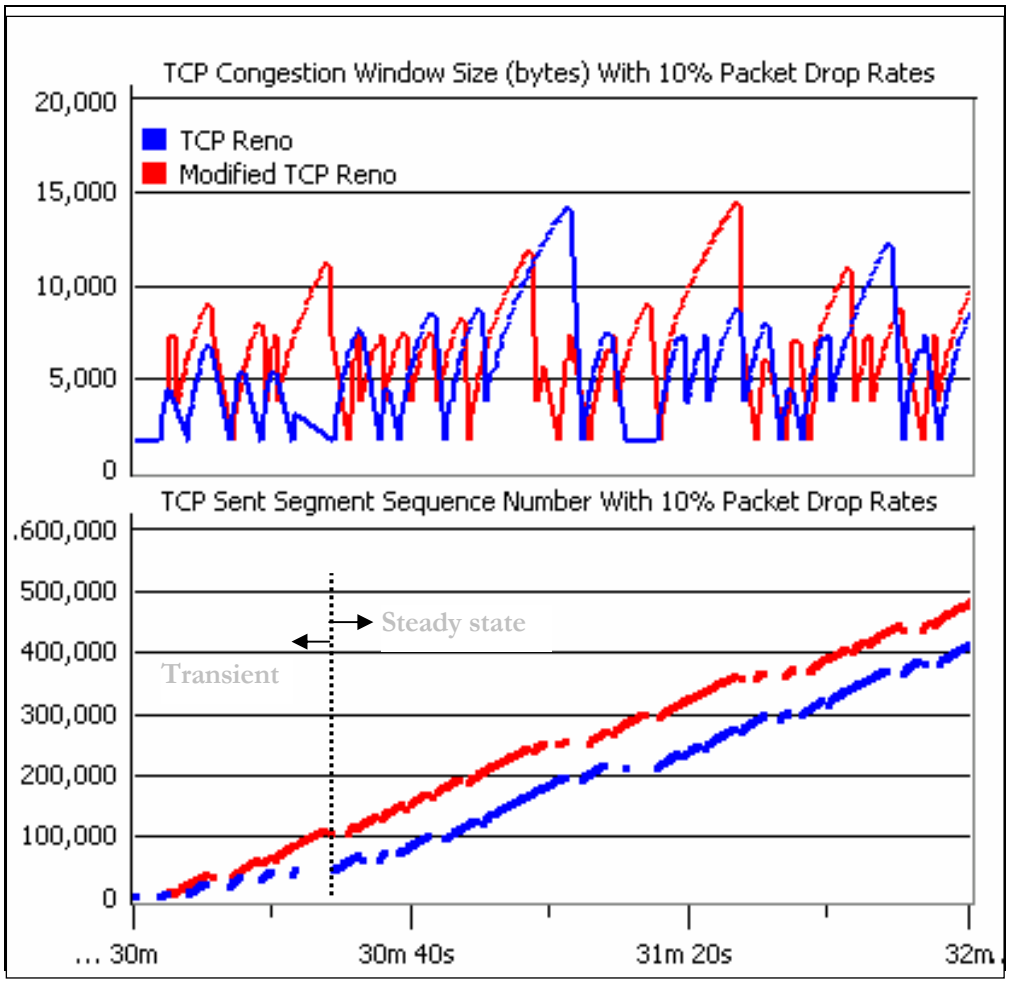

Figure 8-6 TCP CWND and sent segment sequence number responses with 10% packet drop rate

Figure 8.7 compares the steady-state mean TCP CWND size response of our proposed scheme with that of TCP Reno for 10 percent packet drop rates and their mean CWND values are obtained to be 6925 and 6,278 bytes respectively, which closely match with the steady-state CWND value obtained using our analytical model shown in Figure 8.4. Note that the proposed model achieved high throughput performance during the transient phase and maintains its throughput rate during the steady state phase. The reason why it cannot continue to gain throughput improvement is due to the assumptions that receiver window is big enough and does affect the sending rate. However, it achieves the expected CWND improvement as can be seen from Figure 8.7.

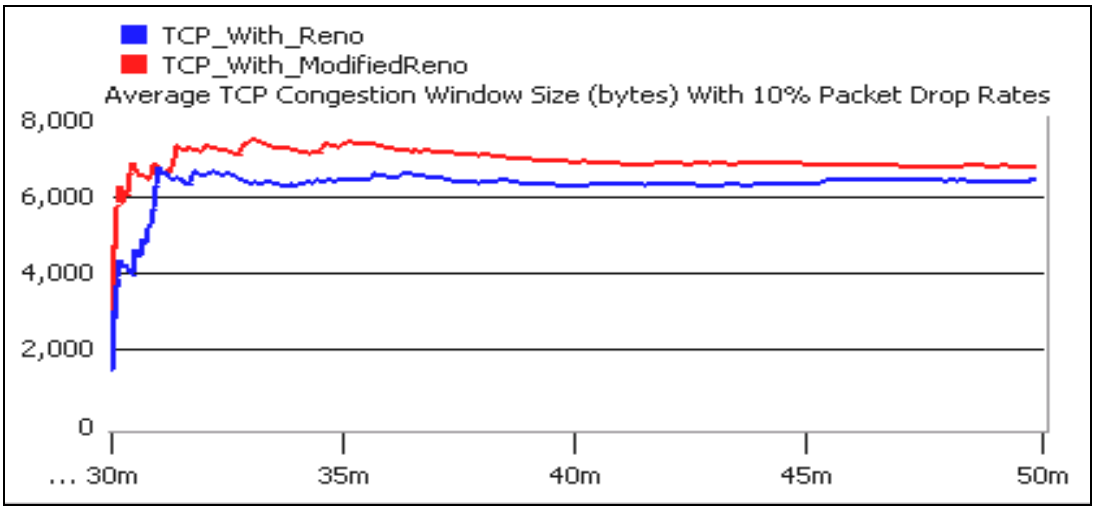

Figure 8-7 TCP cwnd and sent segment sequence number responses with 10% packet drop rate

Since the average CWND with 10% packet drop rates seems to be less than 5MSS, the TCP Reno cannot recover from more than one packet drop within a window of data. This effect can be observed form Figure 8 6. A comparison of the TCP sent segment sequence number responses of the proposed scheme with that of Reno for different packet drop rates is shown in Figure 8.8.

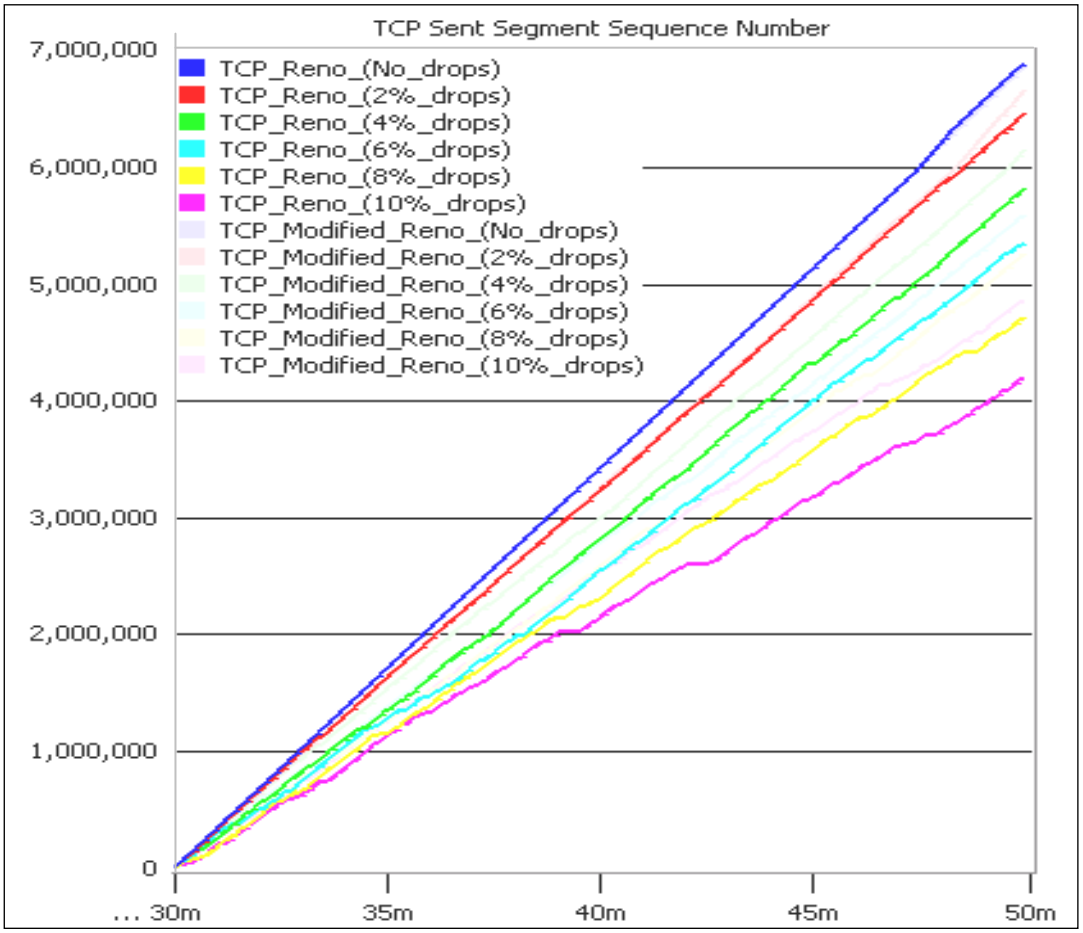

Figure 8-8 TCP sent segment sequence number responses

Figure 8.9 compares the steady-state mean TCP CWND of the proposed scheme with that of both the analytical value, shown in Figure 8.4, and TCP Reno for different packet drop rates. It also shows the TCP CWND improvement with the proposed scheme over TCP Reno. From Figure 8.9, it can be seen that the steady-state CWND value obtained using our analytical model shown in Figure 8.4 closely matches with that of the simulation results.

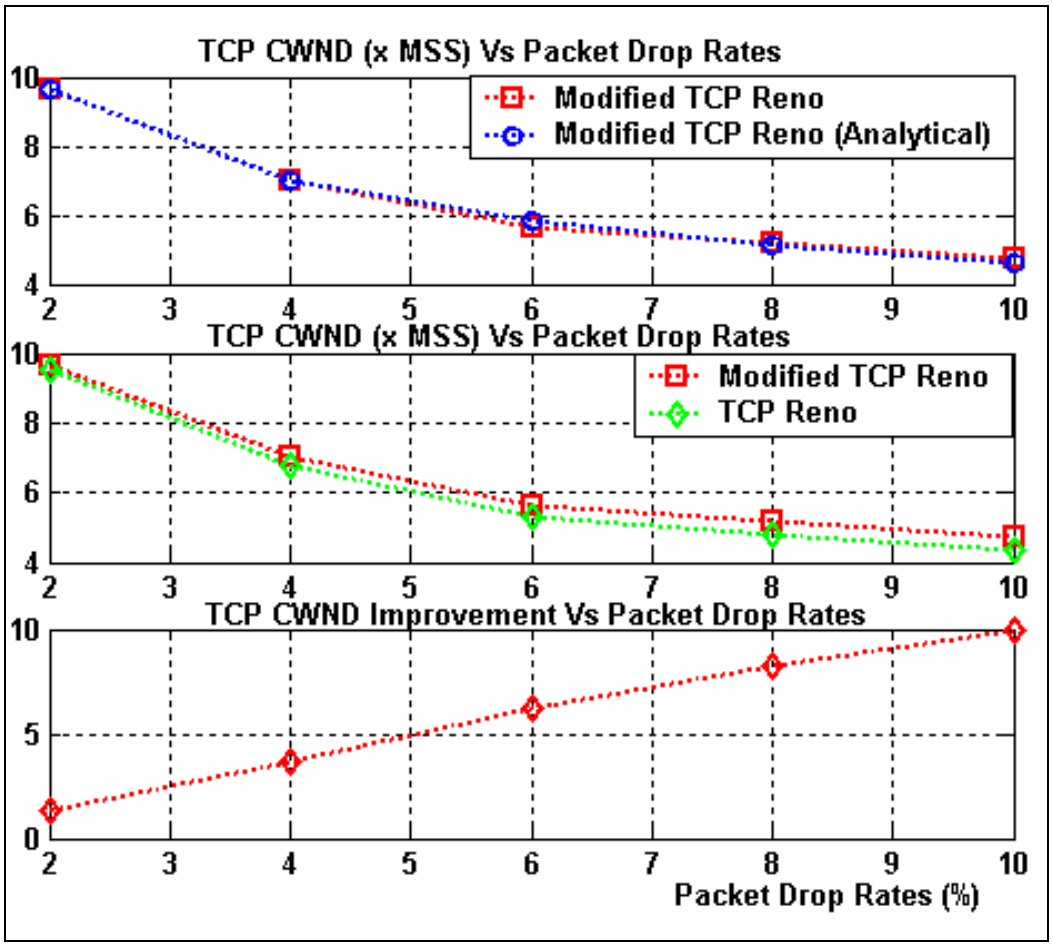

Figure 8-9 Mean TCP CWND and TCP CWND improvement versus packet drop rates (p)

A summary of the average TCP throughput performance with different packet drop rates is given in Table 8.5. From Figure 8.8, 8.9 and Table 8.5, it can be seen that our proposed scheme improved the TCP throughput compared to that of TCP Reno.

TCP Thoughput in (Kbps)	Packet drop rates (%)					
	0	2	4	6	8	10
Reno	45.83	42.59	40.01	36.73	33.67	29.92
Modified Reno	45.71	43.34	41.26	38.52	36.28	32.99

Table 8-5 summary of the average TCP throughput performance

We could not achieve 10 percent TCP throughput improvement with different packet drop rates as expected because the assumptions made in deriving the model, such as the retransmitted packets are not dropped and the CWND is not limited by the receiver's advertised flow control window, do not hold in our simulations studies. The loss of retransmitted packets causes TCP timeouts. These effects can be observed from Figure 8.6.

8.6 CONCLUSION AND FUTURE WORK

A TCP throughput model was developed after modifying the TCP Reno fast retransmit algorithm to avoid timeouts. The model captures the TCP fast retransmit mechanism and expresses the steady state congestion window and throughput as a function of network utilization factor, RTT and loss rate. Based on the new model, a further modification was proposed where the TCP congestion window size is dynamically adjusted, depending on the packet drop rates. This speeds up the packet recovery process and reduces the number of TCP timeouts over networks with heavy packet losses, such as wireless networks. The network utilization factor, derived as a function of packet drop rate, can also be used to provide quality of service.

The proposed model was implemented in OPNET for a UMTS network and its performance was compared with that of TCP Reno. Simulation results showed that the proposed model reduced the TCP timeouts and improved the TCP performance compared to that of TCP Reno. It was found that the model provides a very good match to the steady-state congestion window behavior. The model could not completely avoid timeouts because of the assumption that the packets transmitted during the recovery process are not lost. This assumption can be relaxed and the model can be modified to incorporate the packet loss detection by both three duplicate acknowledgements and timeouts to predict more accurate TCP throughput performance. In future work, we intend to incorporate this, and to further validate the model in other wireless networks such as WiFi and WiMAX.

Conclusion

The main focus of this thesis is the investigation and improvement of the TCP performance over wireless networks, such as IEEE 802.11 WLAN and UMTS networks. Based on a study of the extensive literature on the enhancement of TCP performance over wireless networks, it emphasizes the need to develop techniques to efficiently utilize the available network resources by distinguishing non-congestion related packet losses from the congestion related losses. TCP's inability to distinguish wireless effects from the traditional wire-line effects is the main factor that degrades its performance over networks with wireless links.

In this thesis, we concentrated on two main strategies for enabling the TCP congestion control mechanism to determine the cause for a packet loss. One is the proxy based mechanism that monitors the radio network interface and sends feedback to the source with the status of the wireless link. The other one is based on end-to-end mechanism, in which the packet loss rate is used as the system metric to fine-tune the congestion control mechanism.

The main objective of our proposed proxy based mechanisms is to explicitly inform the TCP source of any effects caused by wireless links while maintaining the end-to-end design philosophy. However, the implementation technique is network dependent. The major contributions of the proxy based schemes are summarized as follows.

- *Development and implementation of RNF feedback mechanism in an 802.11 WLAN network*: A new RNF scheme was developed to detect wireless packet losses and to distinguish them from congestion related packet losses. The base station is equipped with the WLD proxy to detect and notify the TCP sender of the wireless packet losses. TCP Reno was modified to utilize the radio network feedback to distinguish the losses due to wireless effects form the congestion and fine-tuned to perform wireless enhanced fast retransmit and fast recovery mechanism. The RNF scheme was implemented in an 802.11 WLAN model in OPNET. Simulation results showed that the RNF scheme successfully

distinguished the packet losses due to wireless effects form the congestion and improved the TCP performance significantly compared to that of the standard WLAN. It also demonstrated that it can handle multiple TCP connections and utilized the available network resources efficiently and fairly by adapting to the network characteristics.

- *Development and implementation of RNC feedback mechanism in a UMTS network:* The RNC feedback mechanism, similar to the RNF scheme, was developed and implemented in a UMTS network. The GTP layer of the UMTS RNC protocol stack was modified to detect and notify the TCP sender of the wireless packet losses, which is the main difference between the RNF and RNC mechanism. Since the RNC supports multiple Node Bs, the RNC feedback mechanism can be extended to provide further TCP performance enhancement by freezing the TCP sender to handle timeouts caused by handoffs. The simulation results showed that the RNC feedback mechanism significantly improved the TCP performance compared to that of standard TCP over UMTS. Specifically, the scheme recovered from most of the wireless packet losses and minimized the number of TCP timeouts by early triggering of the wireless enhanced fast retransmit algorithm, introduced in TCP Reno. The effect of using TCP Reno with the SACK option was also investigated. It was found that the proposed scheme with SACK option enabled performs better than with the SACK option disabled when the packet drops rate is moderate, otherwise the performances are quite similar

- *Development and implementation of WENP to minimize spurious TCP timeouts in both 802.11 WLAN and UMTS networks:* The WENP scheme was developed to detect both the wireless packet losses and delay spikes in the wireless link, and enable the TCP sender to distinguish them from wireline related packet losses and timeouts. Delay spikes, defined as a sudden and significant change in the RTT, causes spurious TCP timeouts, which have major impact on TCP performance. The WENP proxy is used to detect both the wireless packet losses and the delay spikes. TCP Reno was further modified to utilize the radio network feedback from the WENP to distinguish both packet losses due to wireless effects from congestion and spurious timeouts from normal timeouts. It was also fine-tuned to perform both the wireless enhanced fast retransmit and fast recovery mechanism and the timeout mechanism. This scheme was implemented in both 802.11 WLAN and UMTS networks. The simulation results demonstrated that the proposed

scheme markedly improved the TCP performance compared to that of standard WLAN and UMTS implementations. Particularly, the scheme recovered from most of the wireless packet losses and minimized the number of spurious timeouts by enabling the TCP sender to successfully distinguish wireless effects from congestion related effects. The effect of using modified TCP Reno with the SACK option with the proposed scheme was also investigated. It was found that the proposed scheme with the TCP Reno SACK option enabled does not have any impact on the proposed scheme.

The major advantages of the proposed proxy based mechanisms over the other proxy based schemes, such as Snoop, are:

- Does not add much overhead to the base station since it only caches the TCP header information.

- Does not compete for bandwidth with the TCP sender since it does not perform any local retransmission. It only sends feedback with the ACK packet in the form of control flags.

- Does not violate the end-to-end semantic of TCP.

- Enable TCP sender to completely distinguish wireless packet losses form congestion losses.

- Enable the TCP sender to completely distinguish spurious TCP timeouts form normal timeouts.

- Gives the flexibility to design the TCP congestion control to fine-tune its congestion control mechanism to efficiently utilize the available network resources.

The only drawback of these schemes, like any split connection schemes, is the inability to monitor the radio interface if the IP datagram is encrypted. In this case, the TCP header is inaccessible since the TCP segment, including its header information, is encrypted and cannot be decrypted at the intermediate nodes.

We further developed an end-to-end EPLR scheme by modifying the TCP Reno fast retransmit algorithm to early detect packet losses and to speed up the packet recovery process to

reduce the number of TCP timeouts over networks with heavy packet losses, such as wireless networks. TCP Reno with EPLR scheme was implemented in a UMTS network and its performance was compared with that of Reno and New Reno. Simulation results showed that Reno with EPLR improved the TCP performance and application response time significantly compared to that of both Reno and New Reno by reducing the TCP timeouts, which is the main cause of degradation of the TCP performance in a wireless environment.

Finally, we developed an analytical TCP throughput model with enhanced TCP Reno fast retransmit algorithm to avoid timeouts. The model captures the TCP fast retransmit mechanism and expresses the steady state congestion window and throughput as a function of network utilization factor, RTT and loss rate. Another new feature was added to the proposed model by dynamically adjusting the congestion window size depending on the packet drop rates. This speeds up the packet recovery process and reduces the number of TCP timeouts over networks with heavy packet losses, such as wireless networks. The network utilization factor, derived as a function of packet drop rate, can also be used to provide quality of service.

The proposed model was implemented in OPNET for a UMTS network and its performance was compared with that of TCP Reno. Simulation results showed that the proposed model reduced the TCP timeouts and improved the TCP performance compared to that of TCP Reno. It was found that the model provides a very good match to the steady-state congestion window behavior. The model could not completely avoid timeouts because of the assumption that the packets transmitted during the recovery process are not lost. This assumption can be relaxed and the model can be modified to incorporate the packet loss detection by both three duplicate acknowledgements and timeouts to predict more accurate TCP throughput performance.

Optimizing the TCP performance to react to a packet loss other than congestion remains an open research problem. Although we developed schemes to enhance the TCP performance over wireless link, we have not further studied their effectiveness in mobile environments, i.e. the hand-off effects on our proposed schemes. We believe these problems merit further exploration for finding feasible solutions to make wireless networks even more efficient.

INDEX

References

3GPP. The Third Generation Partnership Project (3GPP). Retrieved August 2, 2007, from http://www.3gpp.org

A. Gurtov. Effect of delays on TCP performance, in Proceedings of IFIP Personal Wireless Communications, Aug. 2001.

A. Gurtov, & Ludwig, R. Responding to spurious timeouts in TCP, in Proceedings of IEEE INFOCOM, March 2003.

Bakre, A., & Badrinath, B. R. (1995). *I-TCP: indirect TCP for mobile hosts.* Paper presented at the Distributed Computing Systems, 1995., Proceedings of the 15th International Conference.

Balakrishnan, H., & Katz, R. H. *Explicit Loss Notification and Wireless Web Performance. Proc. IEEE Globecom Internet Mini-Conference, Nov. 1998.*

Balakrishnan, H., Padmanabhan, V. N., Seshan, S., & Katz, R. H. (1997). A comparison of mechanisms for improving TCP performance over wireless links. *Networking, IEEE/ACM Transactions on, 5*(6), 756-769.

BERTSEKAS, D., & GALLAGER, R. (1996). *Data Networks, Prentice-Hall International, Inc, Englewood Cliffs, N.J., USA, 2nd ed.*

Biaz, S., & Vaidya, N. (1997). TCP over wireless networks using multiple acknowledgements, Texas A&M University, Technical Report 97-001, January

Brakmo, L. S., O'Malley, S. W., & Peterson, L. L. (1994). TCP Vegas: New Techniques for Congestion Detection and Avoidance. *ACM SIGCOMM*, 24-35.

Bruyeron, R., Hemon, B., & Zhang, L. (1988). Experimentations with TCP selective acknowledgment *ACM Computer Communication Review, Vol. 28, No. 2, April, pp. 54–77.*

C. Zhang, & V. Tsaoussidis. (2001). TCP Real: Improving real-time capabilities of TCP over heterogeneous networks. Proceedings of the 11th IEEE/ACM NOSSDAV.

Caceres, R., & Iftode, L. (1995). Improving the performance of reliable transport protocols in mobile computing environments. *Selected Areas in Communications, IEEE Journal on, 13*(5), 850-857.

Cardwell, N., Savage, S., & Anderson, T. (2000). *Modeling TCP latency.* Paper presented at the INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE.

Casetti, C., Geria, M., Lee, S. S., Mascolo, S., & Sanadidi, M. (2000). *TCP with faster recovery.* Paper presented at the MILCOM 2000. 21st Century Military Communications Conference Proceedings.

Cerf, V., & Kahn, R. (1974). A Protocol for Packet Network Intercommunication. *Communications, IEEE Transactions on [legacy, pre - 1988], 22*(5), 637-648.

Chaskar, H., Lakshman, T. V., & Madhow, U. (1996). *On the design of interfaces for TCP/IP over wireless.* Paper presented at the Military Communications Conference, 1996. MILCOM '96, Conference Proceedings, IEEE.

CHIU D. M , & JAIN R. (1989). Analysis of the Increase and Decrease Algorithms for Congestion Avoidance in Computer Networks. *Computer Networks and ISDN Systems, Vol. 17, June , pp. 1–14.*

Chockalingam, A., Zorzi, M., & Tralli, V. (1999). *Wireless TCP performance with link layer FEC/ARQ.*

Cobb, J. A., & Agrawal, P. (1995). *Congestion or corruption? A strategy for efficient wireless TCP sessions.* Paper presented at the Computers and Communications, 1995. Proceedings., IEEE Symposium on.

Dahlman, E., Gudmundson, B., Nilsson, M., & Skold, A. (1998). UMTS/IMT-2000 based on wideband CDMA. *Communications Magazine, IEEE, 36*(9), 70-80.

Dongkyun, K., Hanseok, B., Jeomki, S., & Cano, J. C. (2005). *Analysis of the interaction between TCP variants and routing protocols in MANETs.* Paper presented at the Parallel Processing, 2005. ICPP 2005 Workshops. International Conference Workshops on.

Doshi, B. T., Johri, P. K., Netravali, A. N., & Sabnani, K. K. (1993). Error and flow control performance of a high speed protocol. *Communications, IEEE Transactions on, 41*(5), 707-720.

E. Altman, K. Avrachenkov, & C. Barakat. "A stochastic model of TCP/IP with stationary random losses," ACM Computer Communication Review, vol. 30, no. 4, pp. 231–242, Oct. 2000.

Ender Ayanoglu, Sanjoy Paul, Thomas F. LaPorta, Krishan K. Sabnani, & Gitlin, R. D. (February 1995). AIRMAIL: A Link-Layer Protocol for Wireless Networks.

Fall, K., & Floyd, S. (1996). Simulation-based Comparisons of Tahoe, Reno, and SACK TCP. *ACM Computer Communication Review, Vol 26, No. 3, pp 5-21*(3).

Fieger, A., & Zitterbart, M. (1997). *Evaluation of migration support for indirect transport protocols.* Paper presented at the Global Telecommunications Conference, 1997. GLOBECOM '97., IEEE.

Floyd, S. (1994 October). TCP and Successive Fast Retransmits, Technical report. Retrieved 26 September, 2007, from ftp://ftp.ee.lbl.gov/papers/fastretrans.ps.

Floyd, S., & Fall, K. (1999). Promoting the use of end-to-end congestion control in the Internet. *Networking, IEEE/ACM Transactions on, 7*(4), 458-472.

Floyd, S., Handley, M., Padhye, J., & Widmer, J. (August 2000). *Equation Based Congestion Control for Unicast Applications, Proceedings of the conference on Applications, Technologies, Architectures, and Protocols for Computer Communication* Paper presented at the ACM Press.

FLOYD, S., & HENDERSON, T. (1999). RFC 2582 - The NewReno Modification to TCP's Fast Recovery Algorithm.

Floyd, S., & Jacobson, V. (1993). Random early detection gateways for congestion avoidance. *Networking, IEEE/ACM Transactions on, 1*(4), 397-413.

Fu, S., & Atiquzzaman, M. (2005). *DualRTT: detecting spurious timeouts in wireless mobile environments.* Paper presented at the Performance, Computing, and Communications Conference, 2005. IPCCC 2005. 24th IEEE International.

Gerla, M., Sanadidi, M. Y., Ren, W., Zanella, A., Casetti, C., & Mascolo, S. (2001). *TCP Westwood: congestion window control using bandwidth estimation.* Paper presented at the Global Telecommunications Conference, 2001. GLOBECOM '01. IEEE.

Goel, S., & Sanghi, D. (1998). *Improving TCP performance over wireless links.* Paper presented at the TENCON '98. 1998 IEEE Region 10 International Conference on Global Connectivity in Energy, Computer, Communication and Control.

Goff, T., Moronski, J., Phatak, D. S., & Gupta, V. (2000). *Freeze-TCP: a true end-to-end TCP enhancement mechanism for mobile environments.* Paper presented at the INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE.

Grieco, L. A., & Mascolo, S. (2003). *End-to-end bandwidth estimation algorithms for Westwood TCP congestion control.* Paper presented at the Information Technology Interfaces, 2003. ITI 2003. Proceedings of the 25th International Conference on.

Gurtov, A., Passoja, M., Aalto, O., & Raitola, M. (2002). *Multi-layer protocol tracing in a GPRS network.* Paper presented at the Vehicular Technology Conference, 2002. Proceedings. VTC 2002-Fall. 2002 IEEE 56th.

Haardt, M., Klein, A., Koehn, R., Oestreich, S., Purat, M., Sommer, V., et al. (2000). The TD-CDMA based UTRA TDD mode. *Selected Areas in Communications, IEEE Journal on, 18*(8), 1375-1385.

Haas, Z. J., & Agrawal, P. (1997). *Mobile-TCP: an asymmetric transport protocol design for mobile systems.* Paper presented at the Communications, 1997. ICC 97 Montreal, 'Towards the Knowledge Millennium'. 1997 IEEE International Conference on.

Hengartner, U., Bolliger, J., & Gross, T. (2000). *TCP Vegas revisited.*

Hoe, J. C. (1996). *"Improving the Start-up Behavior of a Congestion Control Scheme for TCP', Proceedings of the ACM SIGCOMM'96, Vol. 26, No. 4, October, pp. 270–280. Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications.*

Jacobson, V. (1990). Modified TCP congestion avoidance algorithm, end2end mailing list (ftp://ftp.isi.edu/end2end-interest-1990.mail), April 30.

Jacobson, V. (1988). Congestion Avoidance and Control. *ACM SIGCOMM Computer Communication Review, pp. 314-329, August 18* (4).

Jacobson, V. (January 1995). Congestion avoidance and control. *ACM SIGCOMM, 25*(1), 157 - 187.

Jamalipour, A. (2003). *The wireless mobile Internet : architectures, protocols and services*. Chichester: Wiley.

Jitendra Padhye, Victor Firoiu y, Don Towsley, & Jim Kurose. Modeling TCP Throughput: A Simple Model and its Empirical Validation, May 30, 1998. *SIGCOM*.

Kassa, D. F., & Wittevrongel, S. (2006). *An analytical model of TCP performance*. Paper presented at the Performance, Computing, and Communications Conference, 2006. IPCCC 2006. 25th IEEE International.

Kent, C., & Mogul, J. (1987). Fragmentation Considered Harmful. *ACM Computer Communication Review, vol. 17, no. 5, Aug.* .

Keshav, S., & Morgan, S. P. (1997). *SMART retransmission: performance with overload and random losses*. Paper presented at the INFOCOM '97. Sixteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE.

Kliazovich, D., & Granelli, F. (2005). *Cross-layer congestion control in multi-hop wireless local area networks*. Paper presented at the Wireless Internet, 2005. Proceedings. First International Conference on.

Korhonen, J., Aalto, O., Gurtov, A., & Lamanen, H. (2001). *Measured performance of GSM, HSCSD and GPRS*. Paper presented at the Communications, 2001. ICC 2001. IEEE International Conference on.

Kuang-Yeh, W., & Tripathi, S. K. (1998). *Mobile-end transport protocol: an alternative to TCP/IP over wireless links*. Paper presented at the INFOCOM '98. Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE.

Kurose, J. F., & Ross, K. W. (2003). *Computer networking : a top-down approach featuring the Internet* (2nd ed.). Boston: Addison-Wesley.

Kurose, J. F., & Ross, K. W. (2005). *Computer networking : a top-down approach featuring the Internet* (3rd ed.). Boston: Pearson/Addison Wesley.

Ladas, C., Amiee, R. M. E., Mahdavi, M., & Manson, G. A. (2002). *Class based selective-ARQ scheme for high performance TCP and UDP over wireless links.* Paper presented at the Mobile and Wireless Communications Network, 2002. 4th International Workshop on.

Leiner, B., Cole, R., Postel, J., & Mills, D. (1985). The DARPA internet protocol suite. *Communications Magazine, IEEE, 23*(3), 29-34.

M. Allman, & V. Paxson. On estimating end-to-end network path properties, in Proceedings of ACM SIGCOMM, Sept. 1999.

Mason, P. C., Cullen, J. M., & Lobley, N. C. (1996). *UMTS architectures.* Paper presented at the Mobile Communications Towards the Next Millenium and Beyond, IEE Colloquium on.

Mathis .M, Mahdavi .J, Floyd .S, & Romanow .A. (April 1996). TCP Selective Acknowledgement Options: RFC 2018.

Mathis, M., Semke, J., & Mahdavi, J. The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm. *Computer Communication Review, 27(3), July 1997.*

Mehta, M., & Vaidya, N. H. (February, 1998). Delayed duplicate acknowledgements: A proposal to improve performance of TCP on wireless links. Tecnical Report (Texas A&M University).

Mellia, M., & Zhang, H. (2002). TCP model for short lived flows. *Communications Letters, IEEE, 6*(2), 85-87.

Mohr, W., & Konhauser, W. (2000). Access network evolution beyond third generation mobile communications. *Communications Magazine, IEEE, 38*(12), 122-133.

Nagle, J. (1984). Congestion control in IP/TCP internetworks. *ACM Computer Communication Review, vol. 14, no. 4, pp. 11–17, Oct. .*

Nitin H. Vaidya, M. N. M. C. E. P. G. M. (2002). Delayed duplicate acknowledgements: a TCP-Unaware approach to improve performance of TCP over wireless. *Wireless Communications and Mobile Computing, 2*(1), 59-70.

O'Hara, B., & Petrick, A. *IEEE 802.11 Handbook : -- a designer's companion*: New York : Standards Information Network, IEEE, c2005.

OPNET. Making Networks and Applications Perform. from http://www.opnet.com/

OPNET Technologies Inc. Technologies Inc, Making Networks and Applications Perform. Retrieved 2 August, 2007, from http://www.opnet.com/

P. Karn, & C. Partridge. (1995). Improving Round-Trip Time Estimates in Reliable Transport Protocols, ACM SIGCOMM, August 1987. *SIGCOMM Comput. Commun. Rev. , V 25*, pp 66-74

Padhye, J., Firoiu, V., Towsley, D. F., & Kurose, J. F. (2000). Modeling TCP Reno performance: a simple model and its empirical validation. *Networking, IEEE/ACM Transactions on, 8*(2), 133-145.

Parsa, C., & Garcia-Luna-Aceves, J. J. (1999). *TULIP: A link-level protocol for improving TCP over wireless links*. Paper presented at the Wireless Communications and Networking Conference, 1999. WCNC. 1999 IEEE.

R. Ludwig, & R. H. Katz. The Eifel algorithm: Making TCP robust against spurious retransmissions. ACM Computer Communication Review, 30(1):30–37, January 2000.

Ramani, R., & Karandikar, A. (2000). *Explicit congestion notification (ECN) in TCP over wireless network*. Paper presented at the Personal Wireless Communications, 2000 IEEE International Conference.

Rendon, J., Casadevall, F., & Carrasco, J. (2002). *Wireless TCP proposals with proxy servers in the GPRS network*. Paper presented at the Personal, Indoor and Mobile Radio Communications, 2002. The 13th IEEE International Symposium.

RFC 761. (1980). Transmission Control Protocol. Retrieved 27 Sept., 2007, from http://www.ietf.org/rfc/rfc761.txt

RFC 793. (1981). Transmission Control Protocol. Retrieved 27 Sept., 2007, from http://www.ietf.org/rfc/rfc793.txt

RFC 896. (1984). Congestion control in IP/TCP internetworks. Retrieved 27 Sept., 2007, from http://www.ietf.org/rfc/rfc896.txt

RFC 1063. (1988). IP MTU Discovery Options. Retrieved 27 Sept., 2007, from http://www.ietf.org/rfc/rfc1063.txt

RFC 1122. (1989). Requirements for Internet Hosts. Retrieved 27 Sept., 2007, from http://www.ietf.org/rfc/rfc1122.txt

RFC 1191. (1990). Path MTU Discovery. Retrieved 27 Sept., 2007, from http://www.ietf.org/rfc/rfc1191.txt

RFC 1323. (1992). TCP Extensions for High Performance. Retrieved 27 Sept., 2007, from http://www.ietf.org/rfc/rfc1323.txt

RFC 2001. (1997). TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms. Retrieved 27 Sept., 2007, from http://www.ietf.org/rfc/rfc2001.txt

RFC 2018. (1996). TCP Selective Acknowledgment Options. Retrieved 27 Sept., 2007, from http://www.ietf.org/rfc/rfc2018.txt

RFC 2309. (1998). Recommendations on Queue Management and Congestion Avoidance in the Internet. Retrieved 27 Sept., 2007, from http://www.ietf.org/rfc/rfc2309.txt

RFC 2481. (1999). A Proposal to add Explicit Congestion Notification (ECN) to IP. Retrieved 27 Sept., 2007, from http://www.ietf.org/rfc/rfc2481.txt

RFC 2507. (1999). IP Header Compression. Retrieved 27 Sept., 2007, from http://www.ietf.org/rfc/rfc2507.txt

RFC 2525. (1999). Known TCP Implementation Problems. Retrieved 27 Sept., 2007, from http://www.ietf.org/rfc/rfc2525.txt

RFC 2581. (1999). TCP Congestion Control. Retrieved 27 Sept., 2007, from http://www.ietf.org/rfc/rfc2581.txt

RFC 2914. (2000). Congestion Control Principles. Retrieved 27 Sept., 2007, from http://www.ietf.org/rfc/rfc2914.txt

RFC 2988. (2000). Computing TCP's Retransmission Timer. Retrieved 27 Sept., 2007, from http://www.ietf.org/rfc/rfc2988.txt

RFC 3390. (2002). Increasing TCP's Initial Window. Retrieved 27 Sept., 2007, from http://www.ietf.org/rfc/rfc3390.txt

RFC 3522. (2003). The Eifel detection algorithm for TCP. Retrieved 27 Sept., 2007, from http://www.ietf.org/rfc/rfc3522.txt

S. Floyd, & Henderson, T. (April 1999). RFC 2582 - The NewReno Modification to TCP's Fast Recovery Algorithm.

Samukic, A. (1998). UMTS universal mobile telecommunications system: development of standards for the third generation. *Vehicular Technology, IEEE Transactions on, 47*(4), 1099-1104.

Seok, S.-J., Youm, S.-K., kim, S.-W., & Kang, C.-H. (2003). A modification of TCP flow control for improving end-to-end TCP performance over networks with wireless links. *Computer Communications, Volume 26*(17), pp. 1998-2010.

Shaojian, F., Atiquzzaman, M., & Ivancic, W. (2002). *Effect of delay spike on SCTP, TCP Reno, and Eifel in a wireless mobile environment.* Paper presented at the Computer Communications and Networks, 2002. Proceedings. Eleventh International Conference.

Sinha, P., Nandagopal, T., Venkitaraman, N., Sivakumar, R., & Bharghavan, V. (1999). *WTCP: a reliable transport protocol for wireless wide-area networks, Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, Seattle, Washington, United States.

Stevens, W. R., & Wright, G. R. (1994). *TCP/IP illustrated* (Vol. 3): Addison-Wesley Pub. Co.

Tanenbaum, A. S. (2003). *Computer networks* (4th ed.). Upper Saddle River, NJ [London]: Prentice Hall PTR ;Pearson Education.

TANENBAUM. A.S. (3rd ed). *(1996), Computer Networks*: Prentice-Hall International, Inc, Upper Saddle River, New Jersey 07458, USA

W.P. Chen, Y.C. Hsiao, J. C. Hou, Y. Ge, & Fitz, M. P. Syndrome: a light-weight approach to improving TCP performance in mobile wireless networks , Communications and Mobile Computing, (2):P 37-57, 2002.

Wenqing, D., & Jamalipour, A. (2001a). *Delay performance of the new explicit loss notification TCP technique for wireless networks.* Paper presented at the Global Telecommunications Conference, 2001. GLOBECOM '01. IEEE.

Wenqing, D., & Jamalipour, A. (2001b). *A new explicit loss notification with acknowledgment for wireless TCP.* Paper presented at the Personal, Indoor and Mobile Radio Communications, 2001 12th IEEE International Symposium.

Wong, J. W. K., & Leung, V. C. M. (1999). Improving end-to-end performance of TCP using link-layer retransmissions over mobile internetworks. *Proc. IEEE ICC'99, 1*, 324-328.

Xie, F., Hammond, J. L., & Noneaker, D. L. (2003). Evaluation of a split-connection mobile transport protocol. *Wirel. Netw. , V 9*(Kluwer Academic Publishers), pp 593-603

Yavatkar, R., & Bhagawat, N. (1994). *Improving end-to-end performance of TCP over mobile internetworks.* Paper presented at the Mobile Computing Systems and Applications, 1994. Proceedings, Workshop.

Yavuz, M., & Khafizov, F. (2002). *TCP over wireless links with variable bandwidth.* Paper presented at the Vehicular Technology Conference, 2002. Proceedings. VTC 2002-Fall. 2002 IEEE 56th.

Zimmermann, H. (1980). OSI Reference Model--The ISO Model of Architecture for Open Systems Interconnection. *Communications, IEEE Transactions on [legacy, pre - 1988], V 28*(4), pp 425-432.