PERFORMANCE OF THE TRANSMISSION CONTROL PROTOCOL (TCP) OVER WIRELESS WITH QUALITY OF SERVICE

By
Tom Walingo

University of Natal
2001

Submitted in fulfillment of the academic requirements for the degree of MScEng in the School of Electrical and Electronic Engineering, University of Natal, 2001.

ABSTRACT

The Transmission Control Protocol (TCP) is the most widely used transport protocol in the Internet. TCP is a reliable transport protocol that is tuned to perform well in wired networks where packet losses are mainly due to congestion. Wireless channels are characterized by losses due to transmission errors and handoffs. TCP interprets these losses as congestion and invokes congestion control mechanisms resulting in degradation of performance. TCP is usually layered over the Internet protocol (IP) at the network layer. IP is not reliable and does not provide for any Quality of Service (QoS). The Internet Engineering Task Force (IETF) has provided two techniques for providing QoS in the Internet. These include Integrated Services (IntServ) and Differentiated Services (DiffServ). IntServ provides flow based quality of service and thus it is not scalable on connections with large flows. DiffServ has grown in popularity since it is scalable. A packet in a DiffServ domain is classified into a class of service according to its contract profile and treated differently by its class. To provide end-to-end QoS there is a strong interaction between the transport protocol and the network protocol. In this dissertation we consider the performance of the TCP over a wireless channel. We study whether the current TCP protocols can deliver the desired quality of service faced with the challenges they have on wireless channel. The dissertation discusses the methods of providing for QoS in the Internet. We derive an analytical model for TCP protocol. It is extended to cater for the wireless channel and then further differentiated services. The model is shown to be accurate when compared to simulation. We then conclude by deducing to what degree you can provide the desired QoS with TCP on a wireless channel.

PREFACE

The research work done in this dissertation was performed by Tom Walingo, under the supervision of Professor Fambirai Takawira, at the School of Electrical and Electronic Engineering, University of Natal, Durban, South Africa. The work was supported by Alcatel Altech Telecom and Telkom South Africa as part of the Center of Excellence Programme at the Center for Radio Access Technologies at the University of Natal.

The work presented in this thesis has been published by the author at two SATNAC conferences.

The whole thesis, unless specifically indicated to the contrary in the text, is the author's work, and has not been submitted in part, or in whole to any other University.

As the candidate's supervisor, I have approved this thesis for submission.

Signed:.....Name:.....Date:.....

ACKNOWLEDGEMENTS

I would like to express my sincere thanks to my supervisor and mentor, Professor Fambirai Takawira for his help and guidance throughout the past two years. I am grateful to have associated with not just a successful researcher but, a truly humble and warm-hearted human being.

Many thanks go to my family, particularly my brother Chiruyi for his encouragement and support during the course of this work.

To my colleague at the center for Radio Access Technologies, I owe you the gratitude for your encouragement and the moments we shared together, including the trying times during the course of this work.

Thanks are also owed to Alcatel Altech Telecomms and Telkom South Africa for their valued financial support for this work.

TABLE OF CONTENTS

3. TRANSPORT CONTROL PROTOCOL AND ITS PERFORMANCE

4. PERFORMANCE IMPROVEMENT OF TCP OVER WIRELESS CHANNELS

5. TCP OVER WIRELESS WITH DIFFERENTIATED SERVICES

6. CONCLUSION

LIST OF FIGURES AND TABLES

Chapter 2

Chapter 3

Chapter 4

Chapter 5

Appendix

LIST OF ACRONYMS

ACK	- Acknowledgement
ADSL	- Asymmetrical Digital Subscriber Loop
AMPS	- Advanced Mobile Phone Service
BER	- Bit Error Rate
BSC	- Base Station Controllers
CDMA	- Code Division Multiple Access
CDPD	- Cellular Digital Packet Data
COA	- Care of Address
CSMA	- Carrier Sense Multiple Access
DBS	- Direct Broadcast Satellite
DiffServ	- Differentiated Services
DRR	- Deficit Round Robin
DSCP	- Differentiated Services Code Point
ECN	- Explicit Congestion Notification
ETSI	- European Telecommunications Standard Institute
FBFQ	- Frame Based Fair Queuing
FER	- Frame Error Rate
FIFO	- First In First OUT
GE	- Gilbert-Elliot
GPRS	- General Packet Radio Systems
GPS	- Generalized Processor Sharing
GSM	- Global System for Mobile Communications
HIPERLAN	- High Performance Radio Local Area Network
IETF	- Internet Engineering Task Force
IntServ	- Integrated Services
IP	- Internet Protocol
LAN	- Local Area Networks
MAC	- Medium Access Protocol
MPLS	- Multiprotocol Label Switching
MSC	- Mobile Switching Centers

NAK	- Negative Acknowledgement
PFQ	- Packet Fair Queuing
PHB	- Per-Hop-Behavior
PQ	- Priority Queuing
QOS	- Quality of Service
RED	- Random Early Detection
RFC	- Request for Comments
RIO	- RED with IN and OUT
RLP	- Radio Link Protocol
RSVP	- Resource Reservation Protocol
RTT	- Round Trip Time
SACK	- Selective Acknowledgement
SCFQ	- Self-Clocked Fair Queuing
SFF	- Smallest Virtual Finish Time First
SLA	- Service Level Agreement
SNR	- Signal to Noise Ratio
SRED	- Stabilized RED
TCP	- Transport Control Protocol
UMTS	- Universal Mobile Telecommunication Systems
VPN	- Virtual Private Networks
WAN	- Wide Area Networks
WAWDN	- Wide Area Wireless Data Networks
WFQ	- Weighted Fair Queuing
WRR	- Weighted Round Robin

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

Due to the rapid advances in the area of wireless communication and the popularity of the Internet, provision of data services like e-mail, web browsing, mobile computing over wireless is gaining importance. It has been shown that 95 percent of the total bytes and 85-95 percent of the total packets on the Internet are of the TCP type [1]. Therefore TCP is the most used transport protocol on the Internet today. With the increase in use and need of mechanisms for delivering audio and video across networks, quality of service (QoS) over networks has received wide attention recently. In this thesis we look at the interaction of the TCP protocol with QoS over a wireless channel. Due to the large volume of TCP traffic in the Internet, gaining a theoretical understanding of the behavior of TCP window adaptation mechanism is an important research goal. The TCP modeling determines the factors influencing the performance of the protocol which gives people working on the protocol the insights on how it can improve. Modeling of the protocol permits network designers to improve the reaction of their networks to incoming TCP packets e.g. at the time of congestion given the current policy of TCP. TCP is facing

several challenges; these include the transmission media, the issue of quality of service and the emerging new protocols among others.

1.2 TCP AND END USER WIRELESS NETWORKS

TCP is a reliable, connection oriented, full duplex, byte-stream, transport layer protocol. It is an end-to-end protocol that supports flow and congestion control. TCP performs flow and congestion control by maintaining a sequence of packets that it can send (window), and by using the slow start, congestion avoidance and first recovery algorithms. When a packet is dropped TCP reacts by reducing its window and therefore wasting some bandwidth. TCP is therefore very sensitive to packet losses. TCP was tuned to perform on wired channels where congestion is the main source of packet loss and not on wireless channels. Currently TCP is supposed to operate on heterogeneous transmission media. The media consist of wired and wireless networks which exhibit different characteristics from each other. In this thesis we look at the performance of TCP over wireless. To understand the need for the study we look at the current trends of the wireless network technologies and the reason why we need to look at TCPs performance.

Wireless networks can be broadly classified as local area networks (LANs) and wide area networks (WANs) depending on the service area of the access point (base station). They provide sufficient bandwidth for office applications but are relatively limited in mobility, typically the user may roam inside a building or campus. There are two main standards for the wireless LANs. The High Performance Radio Local Area Network (HIPERLAN) and the IEEE 802.11 standard also known as wireless Ethernet. The European Telecommunications Standard Institute (ETSI) committee designed the HIPERLAN type 1 (HIPERLAN/1) [64]. HIPERLAN/1 uses Carrier Sense Multiple Access (CSMA) Medium Access Control Protocol (MAC), and can achieve data rates of up to 20 Mb/s in a 50-meter range or up to 1 Mb/s in an 800-meter range. It handles mobile hosts that can be moving at up to 36km/h, and can provide quality of service based on the different categories of data. The standard includes a provision for handoff handling, but does not provide the actual specification for this. HIPERLAN/1 was designed to offer small delays

and is based on small messages that are exchanged relatively frequently. This fact, in combination with its relatively high bandwidth, makes HIPERLAN favorably used for data transfers and other services like teleconferencing, video and medical data transmissions. The original IEEE 802.11 wireless LAN standard [65] was designed to achieve raw bit rates of 1 Mb/s or 2 Mb/s within a 100-meter range. It then developed to 802.11a which offers 40Mb/s in the 5.8GHz band and the 802.11b which offers 11Mb/s in the 2.4GHz band. The 802.11 uses a CSMA with collision avoidance (CSMA/CA) scheme for wireless access. The maximum mobile host roaming speed in 802.11 is 90Km/h.

Wide area wireless data networks (WAWDN) are WANs that are dedicated to data traffic. The current generation of WAWDN includes the cellular digital packet data (CDPD) system and the general packet radio service (GPRS). These packet switched networks allow the user to roam almost everywhere. CDPD shares the channels of the first generation analog cellular network called Advanced Mobile Phone Service (AMPS) that do not carry voice traffic. However, the network operator can specifically assign channels for data traffic. CDPD is based on a CSMA/CD variant called digital sense multiple access and offers IP-based services which is a great advantage. GPRS is normally embedded in a GSM network. It is provisioned in GSM Phase 2+. Its main objective is to provide standard data technologies like TCP/IP with a mobile radio network with significantly higher bit rates than the other systems. It is widely being deployed by European countries but it is seen as a transitional technology towards the third-generation cellular networks. In general the WAWDN provide bit rates of one or two orders of magnitude less than wireless LANs. However, they cover a wide area and have higher roaming speeds.

Cellular networks can handle mixed traffic of voice and data and can also be used for wireless access. They have evolved with time from the analogue first generation cellular networks like AMPS. These are now obsolete due to the limited capacity and services they offer. Second generation cellular networks, like GSM and IS-95 are widely deployed and can be used for wireless network connectivity, though in a circuit switched fashion.

GSM offers bit rates up to 9.6 Kb/s (with GSM Phase 2+ up to 14.4 Kb/s). The GSM designers offer a reliable link layer protocol for data transmission. The forthcoming Enhanced Data GSM Environment (EDGE) will provide speeds up to 560 Kb/s. IS-95 (cdmaOne) cellular standard uses Direct Spectrum Code Division /multiple access (DS-CDMA) and was designed to replace AMPS. IS-95 achieves data rates of up to 19.2Kb/s. Data services can be used simultaneously with voice. Due to the advanced power control needed for CDMA a mobile terminal never uses more transmission power than needed. Due to CDMA's nature, Rayleigh fading and multipath propagation do not degrade the performance significantly. However, cellular networks are not very economical. The user must dial up to achieve connectivity to the network. The users are also charged according to the airtime spent and not the amount of data transmitted or received. Third generation cellular networks like UMTS and 3G WCDMA offer significantly higher data rates. However they have not been fully realized on large scale.

The wireless networks have in overall changed from traditionally providing voice only to a variety of network traffic like data, multimedia and others at higher bit rates. They have also included mobility with increasing roaming speeds and the issue of QoS. These issues have brought challenges to the upper layer protocols. TCP protocol is used in providing these services and this motivates our task in the thesis of understanding the behavior of TCP protocol on a wireless channel.

1.3 THE QUALITY OF SERVICE

TCP does not guarantee any QoS on the Internet. There is a need of providing for QoS on the Internet. In the current Internet there is only one type of traffic class, the best effort. In the best effort traffic class, packets are forwarded when resources are available and none is given preference over the other. The issue of providing QoS in the Internet is under the Internet Engineering Task Force (IETF). They have come up with several QoS architectures which have been written in their work as request for comments (RFCs). In the thesis, the QoS architecture employed is the differentiated services (DiffServ) architecture. The DiffServ architecture was proposed by the IETF and improves on the

current Internet where there is only one type of traffic class. Apart from DiffServ the IETF proposed the Integrated Service model. The Integrated Services together with RSVP signaling attempts to provide per-flow QoS assurances with dynamic resource reservation. Due to its limitation of scalability the IETF has favored the DiffServ which is aimed at traffic aggregates that may not correspond to fine grained flows. The network devices classify packets into aggregates of desired traffic, policing traffic and allocating static or dynamic resources to satisfy QoS requirements. The aim of the DiffServ mechanism is to provide service discrimination in the Internet. It ensures that some traffic is favored over the other. To achieve the aim, DiffServ uses various router mechanisms (buffer management and scheduling mechanisms) to discriminate against packets. The buffer management strategies under investigation employ either randomized packet drops or randomized packet marking techniques to discriminate against various packets in signaling for congestion. From an abstract and idealized standpoint, packet marking and packet dropping are equivalent since they are both congestion indicators. However, they have different effects on transport layer protocols like TCP since packet loss leads to retransmission and associated transient behavior (such as timeout and first recovery). DiffServ can use various queue management strategies which are grouped into active queue management strategies like the Random Early Detection (RED) and passive queue management strategies like drop tail, random drop on full and drop front on full. Various differentiated services class have been proposed; the assured services class [2] and the premium service class [3] are the most popular. The assured service class targets the traffic that has average delay guarantees like email traffic, while the premium service class targets traffic with strict delay guarantee like video traffic.

In terms of QoS, in this thesis, we consider the following areas. Firstly we consider the various buffer management mechanisms and router mechanisms that are effective in providing differentiated services. Secondly we look at the various combinations of the queue management techniques and how they compare to each other on the network. Thirdly we analyze how the assured and premium traffic classes are possible on the network. Since the network is likely to have many traffic models we combine both the assured and premium service models and look at their performance over the channel.

1.4 THE EMERGING PROTOCOLS

TCP has very few expectations on the services provided by the networks and it thus can be run across a large variety of hardware. However, TCP performance is affected by the underlying protocols in terms of how much losses they are introducing in the network. The underlying network layer protocols are also evolving from one technology to the other. As an example we look at the evolution of the Internet protocol.

The Internet protocol provides for transmitting blocks of data called datagrams from sources to destinations. The Internet protocol implements two basic functions. These are addressing and fragmentation, and reassembly of long datagrams, if necessary, for transmission through "small packet" networks. IP is the protocol that hides the underlying physical network by creating a virtual network overview. It is an unreliable, best effort and connectionless packet delivery protocol. The basic IP protocol is the IP Version 4 [66]. However, Ipv4 has been seen to have the following limitations.

- It has a limited address space (32bits) which leads to IP address exhaustion.
- Class routing and network numbering

  The address space for networks is structured into class A, B and C to cater for networks of differing sizes. The space within each class needs to be considered separately. The IP address is divided into a network number and a local part. Once the IP address network number has been allocated, then all addresses within that network are unavailable for allocation elsewhere. Furthermore, the IP addressing model requires that unique network numbers be assigned to all IP networks whether or not they are actually connected to the Internet which is a further waste of numbers.
- The Ipv4 traffic priority class is vaguely defined and scarcely used and it's desirable for modern real time applications.

The limitations of IPv4 have led to its modification into a later version called IP Version 6 [67]. The changes to Ipv6 fall primarily into the categories shown below.

- Expanded addressing capabilities

Ipv6 header format is composed of a 64 bit header followed by two 128-bit Ipv6 addresses for source and destination, for a total length of 40 bytes unlike IPv4 with 32 bit address (See Appendix A1).

- Header format specification

  In IPv6 some header fields (header length field, header checksum, type of service) have been dropped off or made optional. This is for the reason of reducing the common case processing cost of packet handling and to limit the bandwidth cost of the Ipv6 header.

- Improved support for extension and options

  IPv6 incorporates changes in the way IP headers are encoded. This allows for more efficient forwarding, less stringent limits on the length of options and greater flexibility for introducing new options in the future.

- Flow labeling capacity

  In IPv6 a new capability is added to enable the labeling of packets belonging to particular traffic "flows" for which the sender requests special handling, such as non-default quality of service or "real-time" service.

- Authentication and Privacy capabilities

  In IPv6 extensions to support authentication, data integrity, and (optional) data confidentiality are specified.

Current Internet Protocol versions do not support host mobility. Mobile IP [70] represents a simple and scalable global mobility solution. It is not appropriate for supporting fast and seamless handoff control as offered by third generation cellular systems. Mobile IP allows the mobile node to use two IP addresses. The home address (HA), which is static and is used to identify Transport layer connections. The other is the Care of Address (COA) which indicates the point of attachment of the mobile node with respect to the network topology. Cellular IP [71] inherits cellular system principles for mobility management, passive connectivity and handoff control, but is designed based on an IP paradigm. Cellular IP supports local mobility i.e. mobility inside an access network. To provide global mobility support, mobile IP should be used in conjunction with cellular IP. The universal component of the cellular IP network is the base station which serves as a

wireless access point but at the same time routes IP packets and integrates cellular control functionality traditionally found in mobile switching centers (MSC) and the base station controllers (BSC).

TCP, which operates on top of IP, has to cope with the changes of the underlying protocols. The mobile oriented IP protocols will further introduce more losses to TCP due to the numerous handovers and the handover time which can cause a timeout. The IP layer can be modified to help in combating the losses. In the thesis we modified the IP layer to incorporate an intelligent agent for caching packets and used it to improve the performance of TCP as the Snoop protocol.

1.5 MOTIVATION AND FOCUSS OF THESIS

There is a considerable amount of research that has been done on TCP. Most of this work is based on simulations. Given the closed loop control present in the TCP adaptation scheme an accurate analysis is required for modeling window evolution of each flow explicitly as a Markovian stochastic process and the determination of the aggregate behavior of the aggregate arrival process. The transmission protocol has quite a number of parameters it maintains in its operation; these include the slow start threshold, the retransmission time, the timeout value and many others. Given its importance we need an accurate analytical model that can evaluate its performance with respect to these parameters. Most of the models for simulation and analysis of TCP run short in many aspects. [36] doesn't analyze various versions of TCP, [26] doesn't capture the various TCP phenomena like loss recovery, [37] doesn't capture phenomena like the window limitation, [33] doesn't model fast recovery. And thus you can hardly find a model that incorporates almost all aspects of TCP. In the thesis we develop an accurate analytical model for analyzing various TCP parameters. In our model we can incorporate almost all parameters of TCP that we need to analyze.

TCP operates on wireless channels with losses. Traditionally the wireless channel losses have been modeled with a fixed loss probability [26][30][34]. In these models a packet is

lost with a fixed probability that is independent of any other packets lost. However, we argue that even though the TCP protocol operates high up in the OSI protocol stack, due to the correlation of the channel, a packet in the next slot is affected by the channel conditions in the previous slot. The effect of the correlated loss channel filters through the stack to the TCP layer. This issue of correlation between successive packets has been observed in [33][35]. So we study the behavior of TCP over a correlated loss channel. The correlated loss channel is modeled as the Gillbert Elliott (GE) model.

Over the last decades TCP's well-known congestion control algorithm has supported cooperative and end-to-end congestion control on the Internet. It was primarily designed to operate on environments where packet loss was the only available indicator of network congestion and where queues employed the drop tail strategy. The advent of more sophisticated congestion indicators like the Explicit Congestion Notification (ECN) provides us a significant opportunity to modify the existing TCP response to remove certain drawbacks associated with the current window adaptation algorithm. To that extend it is important to consider a more generalized form of TCP window adaptation and to understand how possible changes in the window adaptation parameters might affect the performance of TCP flows. In this case we looked at the behavior of the ECN end-to-end mechanism over a wireless channel.

The issue of QoS is also under intense scrutiny in the research field. Like TCP not much analytical work has been done. [15] looks at the Resource Reservation protocol for the integrated service. However, most work is directed towards DiffServ. In [68] they analyze various router and packet mechanisms to satisfy session requirements. [56] deals with quantifying the throughput while [22] considers relative and absolute differentiated services model. There has been little analytical and simulation results for TCP with differentiated services. In [56] they deal with TCP throughput guarantees in a DiffServ network but they don't consider the wireless channel and the packet is lost with a fixed probability. Therefore combining TCP over wireless with differentiated services is a new field both by simulation and analysis. In this thesis we therefore undertake the task of looking at TCP over wireless with DiffServ both by analysis and simulation.

Motivated by the considerations mentioned in the introductory section above we concentrate on investigating the following problems in the thesis.

- We model the assured, premium traffic and the combined DiffServ schemes and understand how they behave in the Internet. We determine whether these traffic types can be offered on the Internet. We relate the different router mechanisms that can be used to achieve the different service classes and how they perform.
- We develop an analytical model for TCP and use it to understand the possible behavior of the TCP window adjustment algorithm with respect to parameters like the retransmission time, timeout time and packet loss probability.
- We develop an analytical model for TCP over wireless which can be adapted to incorporate as many aspects of TCP as possible. We study the ways of improving TCP performance degradation over wireless.
- We develop an analytical technique to derive the congestion window distribution, throughput and other statistics when multiple TCP flows interact with RED buffers which discriminate against different traffic. The technique requires solving for the window distribution and therefore throughput of a single TCP flow subject to congestion indication with a variable state dependent probability.

1.6 THESES ORGANIZATION

After the general view of the objective of the thesis in this chapter we now consider the rest of the thesis. In Chapter 2 we present the introduction of QoS in the Internet at the network layer. We have chosen IP at the network layer due to its growing popularity. We consider the mechanisms that can be used to provide different QoS in the Internet. We then focus on providing for QoS on the Internet. We focus on the differentiated services approach and find out if the specified traffic classes can be achieved on the Internet.

In Chapter 3 we consider the TCP window algorithm. We develop our own analytical model of TCP. Using our model, the few models in literature and simulations we analyze the performance of the TCP protocol. We establish the effect of varying TCP algorithm

parameters on its performance. We check how our model performs in comparison to other models and simulations.

In Chapter 4 we develop the analytical model for TCP over wireless that helps us analyze the performance of TCP algorithm over wireless channels. We then consider the ways of improving TCP performance over wireless channels. In particular we focus on the two important methods of improving TCP over wireless, the Snoop Protocol and our modified Explicit Congestion Notification protocol and compare their performance.

In Chapter 5 we combine the performance of TCP over wireless with differentiated services. We look at the extend to which we achieve QoS in a wireless network given that TCP does not perform well on the wireless network.

Finally in Chapter 6 we summarize our work on TCP over wireless with differentiated services. We indicated further work that is required in the area of TCP over wireless with DiffServ.

1.7 CONTRIBUTION OF THESIS

Ideas in the various aspect of the Internet are given and documented in the RFC's, but most of them have not been tested to see if they are possible or not. In this thesis we have actually shown that the following proposed ideas are realizable.

- RFC 2481 [54] proposes the addition of ECN to IP. This is a different form of congestion notification from packet loss. We modify ECN for a wireless channel and show that the TCP Performance is improved with the ECN.
- RFC 2597 [19] proposes an assured forwarding behavior for the Internet. We come up with results to show that this class of service is possible in the Internet.
- RFC 2638 [3] proposes combining assured and premium traffic. We analyze the possibility of the coexistence of the two service classes and come up with results.
- RFC 2488 [50] summarizes some methods of improvement of TCP over wireless. We analyze the performance of some of the methods, these include the various TCP protocol improvements and the snoop protocol.

TCP over Wireless with differentiated services is a new area of research and not much work has been done. We have hardly come across a paper that combines the performance of TCP over wireless with differentiated services. We have developed an analytical framework for analyzing TCP over wireless with differentiated services. Our analytical framework has the following merits.

- It can be used to model many sources in the network. The few analytical formulas developed [27] [28] can only model one source on the network.
- The analytical model is adaptive e.g. it can be used to model plain TCP over wireless.
- The analytical model can be extended to any of the many version of TCP.
- TCP has many parameters that affect its performance. In the model you can incorporate many TCP parameters as possible. The only limitation in this respect might be the protocol complexity.

The work done in the thesis resulted in the following publications:

- Tom Walingo and Fambirai Takawira, "Differentiated Services over GPRS," SATNAC 2000 conference, Cape Town, South Africa. [21]
- Tom Walingo and Fambirai Takawira, "Comparison of Snoop and ECN Protocols," SATNAC 2001 conference, Wild Coast Sun, South Africa. [69]
- Tom Walingo and Fambirai Takawira, "TCP Over Wireless With Differentiated Services," To be submitted to IEEE/ACM Transactions on Networking Journal.

CHAPTER 2
QUALITY OF SERVICE IN THE INTERNET

2.1 INTRODUCTION

Congestion in the network is the state of sustained network overload where the demand for the network resources is close to or exceeds capacity. The network buffers are limited in size and drop packets during congestion when they exceed their capacity. One of the ways of alleviating congestion and dropping of packets in the network would be having very large buffer sizes. However, it has been shown that even if the networks buffer sizes were to be very large there would still be a problem since the packet lifetime in the queues would expire and they would be thrown away by the appropriate protocols [4]. Infinite queues would encourage higher delays that are undesirable for real time applications. To reduce the buffer sizes and introduce QoS in the Internet we need some form of buffer management. The buffer management alone might not be sufficient and it is normally combined with scheduling mechanisms. In fact the whole idea of QoS revolves around the management of the router buffers and scheduling mechanisms. QoS

measures are mostly applied at the router (IP layer). This is because the Internet router is the best candidate for detecting congestion and taking appropriate action like dropping the packet. In this chapter we look at the mechanisms that can be used in the Internet buffers to provide quality of service. We then focus on the IETF QoS schemes. We present the DiffServ's traffic classes (assured and premium) and determine whether these traffic classes can be achieved and what happens when we combine the two traffic models on the same network. We then consider whether the achieved quality of service is absolute or it is relative between two traffic types of different reservation.

2.2 QUEUE MANAGEMENT AND CONTROL IN THE INTERNET

Queue management and control involves the router mechanisms that help in achieving the quality of service in the Internet. The router mechanisms include buffer management strategies, queue management and packet scheduling algorithms.

2.2.1 Buffer Management

The role of buffer management is to determine how the buffer space is shared between the different flows that traverse a gateway, particularly those flows that use the same output interface. The most popular buffer management schemes are shared buffer pool and per flow allocation. The strategies of buffer allocation can vary, they can be static, or they can be dynamic and based on different criteria such as number of flows, current or past bandwidth allocation, and buffer occupancy. In shared buffer pool, buffers are on first come first use basis, and there is no clear protection between flows since one flow can occupy all the buffers and starve all the other flows by simply sending fast enough. This scheme is simple and efficient to implement hence it is found in most Internet routers. The per flow allocation protects flows from each other by keeping track of buffer utilization and dropping packets based on buffer occupancy level of each flow. The scheme is expensive and cannot scale in terms of processing power to meet the requirements of large number of flows in the backbone routers. Buffer management is mostly combined with the queue management strategies as we see in the next section.

2.2.2 Queue Management

The role of queue management is to control the length of the queue and potentially which flows occupy it, by selecting which packets to drop and determining when this is appropriate. Queue management strategies are complementary to both scheduling (which determines the service order) and the buffer management (which determines the number of queues per output interface). Queue management algorithms provide feedback to the source by dropping the packets or marking them. The queue management types are classified according to whether they are for congestion recovery (when there is extreme congestion) or congestion avoidance. Congestion avoidance algorithms maintain the queue in a region of low delay and high throughput. Based on the congestion level the queue management types fall into two, the passive queue management algorithms which only apply during congestion recovery and the active queue management algorithms which apply during congestion avoidance.

2.2.2.1 *Passive queue management*

This is the traditional way of queue management. It involves setting a maximum length for each queue and accepting packets until the queue is full. When the queue is full some packets are dropped until the queue decreases after a packet from the queue has been served. They are classified further according to the way they drop packets when the queue is full into the following types.

- Tail drop: The router accepts packets until the queue is full and then drops the subsequent incoming packet.
- Random drop on full: The router accepts packets until the queue is full and then drops a randomly selected packet from the queue when a new packet arrives.
- Drop front on full: The router accepts packets until the queue is full and then drops the packet at the front of the queue when a new packet arrives

The passive queue management strategies have several problems [5], some of the problems include:

- Lockout: In some cases tail drop allows a single or few connection(s) to monopolize queue space preventing other connections from getting room in the queue. The lockout phenomenon is often a result of synchronization or other timing effects.
- Full queues: They allow queues to maintain full status for long time periods. Congestion is only signaled when the queues are full. These eventually results in a drawback for traffic requiring low end to end delay in the network. Packets often arrive at the queues in bursts and thus full queues will lead to multiple packets being dropped, these can result in global synchronization of flows throttling back followed by a sustained period of lower link utilization and thus the overall throughput is reduced.

The aim of buffering is to absorb data bursts and transmit them during appropriate periods. This leads to the need for small queues to provide the capacity to absorb data bursts and thus queue limits should reflect the size of the bursts to be absorbed and not the steady state queues. Random drop and drop front on full solve the lockout problem of drop tail. However, they don't solve the full queues problem and they are expensive in operation due to sorting of the queues in order to drop a packet.

2.2.2.2 Active queue management

The solution to the full queues problem is to allow buffers to drop packets before the queue is full and thus the end nodes respond to congestion before the buffer overflows. This is called active queue management. Active queue management aims to achieve the following.

- Active queue management aims at reducing the number of packets dropped in routers as a result of buffer overflow. During buffer overflow more packets are dropped in the routers. These results in several effects. Firstly, like in the case of tail drop, it results in global synchronization (phenomena of many connections reducing their traffic at the same time) which results in lower average link utilization. Secondly if used with TCP, TCP recovers with more difficulty from a burst of losses than a single loss. Thirdly, unnecessary packet drops represent a possible waste of bandwidth on the way to the drop point.

- Active queue management aims at providing lower delay interactive service by keeping the queue size small. This is important for interactive applications such as short web transfers, Telnet traffic, or interactive audio-video sessions, whose subjective performance is better when the end-to-end delay is low.
- Active queue management avoids lockout behavior by ensuring that there is always an available buffer available for an incoming packet. It also prevents router bias against low bandwidth but highly bursty flows. Lockout results in unfairness among flows.

2.2.2.3 Random Early Drop (RED)

The RED [6] algorithm is an active queue management algorithm whose goals are to provide congestion avoidance by controlling the average queue size, to avoid global synchronization, avoid the bias against bursty traffic and have the ability to maintain an upper bound on the average queue size even in the absence of cooperation from transport layer protocols. The RED algorithm consists of two separate algorithms, the algorithm for computing the average queue size and the algorithm for packet marking.

2.2.2.3.1 Packet Marking

In this algorithm RED decides whether or not to mark an incoming packet. Marking a packet can be by dropping a packet or setting a bit in the packet header or any other method understood by the transport protocol. TCP uses the packet drop marking. RED maintains two parameters, a minimum threshold ($\min_{th}$) and a maximum threshold ($\max_{th}$). When the average queue (avg) is less than the minimum threshold, no packets are marked and when the average queue is greater than the maximum threshold, every arriving packet is marked. When the average queue size is between the maximum and minimum threshold, each arriving packet is marked with a probability P_a which is a function of the average queue size. Each time a packet is marked the probability that a packet is marked from a particular connection is roughly proportional to the connections

share of bandwidth at the gateway. As avg varies from $\min_{th}$ to $\max_{th}$, the packet marking probability P_a varies linearly from 0 to $P_{\max}$.

$$P_a = P_{\max}(avg - \min_{th})/(\max_{th} - \min_{th}) \quad (2.1)$$

Figure 2.1 illustrates the working of the RED algorithm, the x axis is the average queue size and the y axis is the probability of dropping a packet.

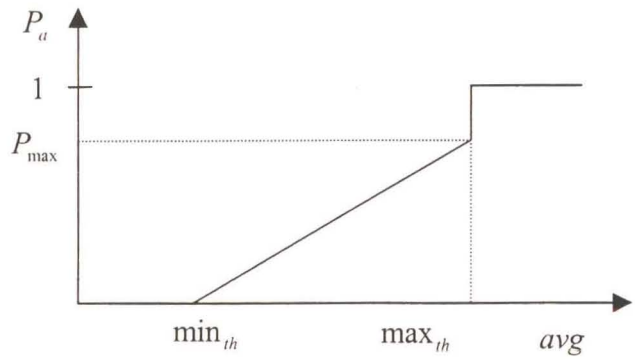


Figure 2.1. RED operation

The choice of $\min_{th}$ and $\max_{th}$ depends on the desired average queue size. For highly bursty traffic $\min_{th}$ should be fairly large to allow the link utilization to be maintained at an acceptably high level. The optimal value for $\max_{th}$ depends on the maximum delay that can be allowed by the gateway. The marking probability on the other hand can be according to any random distribution as deemed fit, say geometric or uniform random distributions.

2.2.2.3.2 Estimation of average queue size

The RED gateway uses a low pass filter to calculate the average queue size. The gateways calculation of average queue size takes into account the period when the queue is empty (idle period). It estimates the number m of small packets that could have been transmitted by the gateway during the idle period. After the idle period the gateway computes the average queue size as if m packets had arrived to an empty queue during that period. With the low-pass filter a short-term increase in the queue size (q) that results from bursty traffic or transient congestion do not result in a significant increase in

the average queue size. The low pass filter is an exponential weighted moving average which is given by

$$avg(t+\delta) = [1 - w_q]avg(t) + qw_q \tag{2.2}$$

where $avg(t)$ is average queue length at time t and $avg(t+\delta)$ is the average queue length at a small time δ after t, w_q is a weight that determines the time constant of a low-pass filter. This weight is determined by the size and duration of bursts in queue size that are allowed by the gateway. As shown in [6], if we want to absorb bursts of length L the upper bound of w_q is chosen to satisfy the following equation

$$L + 1 + \frac{(1 - w_q)^{L+1} - 1}{w_q} < \min_{th} \tag{2.3}$$

The choice of the lower bound is however flexible and it depends on how you want the average size to reflect the changes to queue size. The parameter w_q determines how fast your router responds to the instantaneous changes in the queue size. If w_q is set too low it responds too slowly to the changing queue size and the gateway might not be able to detect the initial stages of congestion.

Unlike the drop tail with one parameter (buffer size) the RED gateway has extra parameters as seen above, these are $\min_{th}$, $\max_{th}$, $P_{\max}$ and w_q. Congestion avoidance algorithms are required to have low parameter sensitivity. The parameters should be applicable to networks with widely varying bandwidths. There are some rules that can give a guideline in the selection of the parameters [6] and they are as below.

- Ensure adequate calculation of the average queue size: Since w_q gives an indication of how the average queue size reflects the real queue size it should not be set too low for the calculated average queue length to delay for too long in reflecting increase in actual queue length. As a rule of the thumb set $w_q \geq 0.001$.
- Set $\min_{th}$ sufficiently high to maximize network utilization.
- Make $\min_{th} - \max_{th}$ sufficiently large to avoid global synchronization since the gateway will mark less packets at a time. The rule of the thumb is to set $\max_{th}$ to at

least twice $\min_{th}$. If $\min_{th} - \max_{th}$ is too small, then the computed average queue size can regularly oscillate up to $\max_{th}$. This behavior is similar to the oscillations of the queue up to a maximum queue size with Drop Tail gateways.

Though currently superior to other algorithms the RED mechanism has its own weaknesses. Firstly it is difficult to parameterize RED queues to perform well under different congestion scenarios. Secondly congestion notification does not directly depend upon the number of connections multiplexed over the link [7]. Thirdly, the RED algorithms benefits decrease significantly when packet loss is used as a means of congestion notification. These weaknesses have led to ECN and an adaptive RED algorithm like the Self-Configuring RED [7]. Self-Configuring RED infers whether RED should become more or less aggressive by observing the variations in the average queue length. It has been shown to perform better than the normal RED in some cases. The other RED algorithms variant is the Stabilized RED (SRED) [8]. SRED like RED preemptively discards packets with a load dependent probability when the router buffer seems to be congested. It has the additional feature which helps it stabilize its buffer occupation at a level independent on the number of active queue connections. This is done at a wide range of load levels. Another RED algorithms problem is using queue length as an indicator of severity of congestion, as an example one source with large number of packets is just the same as many different sources in RED. A different queue management algorithm BLUE [9] has been proposed that uses packet losses and link idle events to manage congestion rather than on the instantaneous and average queue length. Most of the variants of RED are still under investigation and haven't been implemented yet. So the ordinary RED is still the most popular congestion algorithm to be deployed in the Internet and hence it's the one that will be used in the thesis.

2.2.3 Scheduling Algorithms For The Internet

The role of a scheduler in a router is primarily to decide in what order service requests (can be incoming packets) are allowed access to resources (output queues and output lines). Another role of a scheduler is to manage service queues (output buffers).

Schedulers determine which packet to send next and are primarily used to manage the allocation of the limited bandwidth. The scheduler implements its policy through an algorithm called a scheduling discipline. Figure 2.2 shows a router output interface indicating the operation of a scheduler.

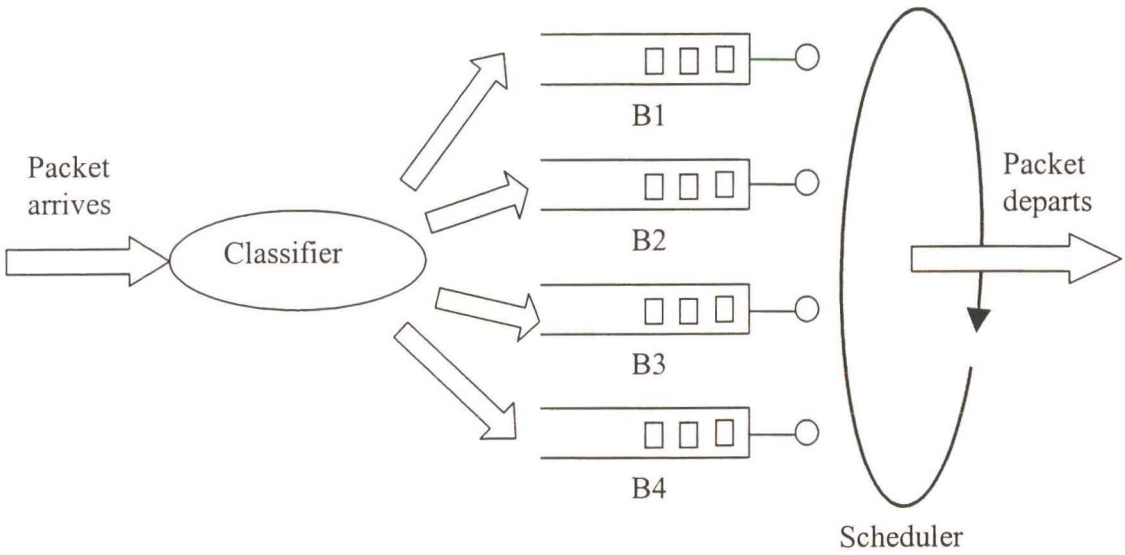


Figure 2.2 Router output interface

The main requirements for scheduling discipline include [23]:

- Ease of implementation

A scheduling system with a less complex algorithm and one which requires less state information is easier to implement in hardware and suitable for high-speed networking.

- Fairness and protection

A network with different traffic sources requires that certain bandwidth should be guaranteed for each traffic source. The protection requirement requires that no flow should suffer due to the misbehavior or characteristics of any other flow. The max min fairness criterion [10] allows precise allocation of resources by allowing per hop local fairness for each flow. This results in global fairness for each flow. The demands of the resources flow are ordered in increasing demand. Flows with lowest demand are allocated resources first since the small demands are likely to be all that they ask for. The actual resource allocation to flow n, m_n and the resources available to a flow n, M_n are as in equation 2.5 below.

$$m_n = \min(x_n, M_n) \qquad 1 \le n \le N \tag{2.5}$$

$$M_n = \frac{C - \sum_{i=1}^{n-1} m_i}{N - n + 1}$$

where C is the maximum resource and x_n is the resource demanded by flow n, $x_1 \le x_2 \ldots \le x_n$. Weights can be assigned to the resource demands so that some demands receive a greater share of the resources than others.

- Performance bounds

To support QoS in networks, it should be possible to specify performance bounds to ensure the network handles the flow appropriately. If a scheduler is providing a guaranteed service, it must be able to work with deterministic performance bounds. If relative priorities for the flow were being specified, then the scheduler should allow performance bounds to be specified statistically or probabilistically.

- Admission control

In case deterministic performance bounds are specified, the network might need to perform admission control before it begins flow transmission.

Scheduling algorithms are generally divided into two groups: Work conserving and non-work conserving. For a work conserving scheduling policy, the router is never idle when packets are waiting to be serviced and giving one flow a lower delay or higher data rate in a work conserving scheduler means reducing the data rate of the other flow. A non-work-conserving scheduler can be idle even if packets are waiting to be served. This is because the scheduler waits for packets to become eligible for transmission. Packet transmission eligibility is determined in several ways. This can be to ensure that the packet always spends a fixed time in the router or to establish a fixed end-to-end delay for a packet. The main advantage of non-work-conserving disciplines is that they reduce jitter, making downstream traffic more predictable at the cost of higher end-to-end delay. Their main disadvantage is their implementation complexity. Some of the Scheduling algorithms are discussed in the following sections.

2.2.3.1 First In First Out (FIFO, FCFS)

FIFO is the traditional IP router scheduling discipline. In its simplest form, FIFO queuing involves storing packets when the network is congested and forwarding them in order of arrival when the network is no longer congested. FIFO mechanism is relatively easy to implement and offers high cost efficiency since the output link and buffer capacity are utilized very efficiently. The main disadvantage of FIFO is the lack of quality differentiation since the packets going through a FIFO queue are treated the same way and hence every session experiences the same delay and packet loss characteristics. FIFO queue has little impact on traffic control and therefore relies on the traffic control functions in boundary nodes thus making the system vulnerable to misbehaving users. The main concern of most FIFO scheduled routers is queue management of excessive packet burst and finite buffer space. Bursty sources can cause long delays in delivering time-sensitive application traffic, and potentially to network control and signaling messages. The FIFO system has been seen to have increased jitter as the number of hops in a network increases. This is due to the fact that the packet has a separate opportunity for uncorrelated queuing delays at each hop. A modified FIFO called FIFO+ [11] has been introduced to reduce this problem. It aims at correlating the sharing experience a packet has at the successive nodes in its path.

2.2.3.2 Priority Scheduling (PQ)

In PQ a number of distinct queues are created and a level of priority is associated to each one of them. The packets are classified to one of the priority queues. The individual queues are served in a FIFO scheduling discipline. A packet in the higher priority is served only when all the queue of a higher priority are empty. Busy higher priority queues could thus prevent lower priority queues from being served, a situation known as *starvation*. The higher priority queues thus end up having the lowest delay and highest throughput and bandwidth. The end-to-end performance guarantees are not provided per flow basis but by classes. A disadvantage of PQ is that it uses static configuration and thus does not automatically adapt to changing network requirements. Priority queuing can flexibly prioritize traffic according to network protocol (for example IP, IPX, or

AppleTalk), incoming interface, packet size, source/destination address, and so on. PQ is useful for making sure that mission-critical traffic traversing various WAN links gets priority treatment.

2.2.3.3 Round Robin

In Round Robin a small unit of time, called timeslice or quantum, is defined. The scheduler goes around the queues in a round robin fashion (in a prescribed order), transmitting at least a packet in each queue for a time interval of one quantum. Normally each queue is visited once per round. Extensions to Round robin scheduling include:

- The Weighted Round Robin (WRR)

A number of service classes are defined, and to each service class a weight is assigned. Thus WRR supports a number of priority levels, the number of sequential time slices that each service class can get during its service turn depends on the weight of the service class. The available service opportunities are distributed to all service classes in proportion to their weight factors with classes having larger weight factors getting serviced more often than those with smaller weight factors. In WRR low priority classes regardless of the traffic volume of high priority classes are guaranteed minimum service slots. Due to this factor various QoS requirements are met. In WRR a time quantum is defined in terms of the number of packets. One packet is served in one time slot (quantum). The assigned weights are normalized by dividing them with the average packet size for each flow. So when the packet sizes vary the weights might become irrelevant. WRR is suitable for networks with fixed packet sizes.

- Deficit Round Robin (DRR)

DRR solves the problem of the requirement to know the average packet size for each flow. In DRR each nonempty queue has a deficit counter that begins at zero and it is also set to zero when the queue is empty after service. As the scheduler visits each nonempty queue, it reads the packets at the head of the queue and tries to serve one quantum of data. A packet at the head of the queue is served if it is less than or equal to the sum of the quantum size and the deficit counter. If the packet cannot be served then the value of the deficit counter is increased by the quantum size for that queue.

2.2.3.4 Fair Queuing Algorithms

Generalized Processor Sharing (GPS) [12] is a work conserving scheduler that provides max-min fairness and flow protection. GPS guarantees an end-to-end QoS to a session whose traffic is constrained [10], say by a leaky bucket. This property is the basis for supporting guaranteed service in the network. GPS ensures fair allocation of bandwidth among all backlogged sessions in the network regardless of whether or not their traffic is constrained. This property is the basis for supporting best-effort service in networks. The GPS server has N queues associated with a service share. During a service interval when there are exactly M non-empty queues, the server serves the M packets at the head of the queue simultaneously, in proportion to their service shares. GPS provides a simple model for supporting integrated services networks, but it cannot be implemented precisely.

To approximate GPS, packet fair queuing (PFQ) algorithms have been proposed. A PFQ algorithm maintains three parameters for each session; system virtual time, virtual start time and the virtual finish time. All packet fair queuing (PFQ) algorithms are packet-approximations for the GPS discipline. They all use a similar priority queue mechanism based on the notion of a virtual time function. They differ in two aspects, the virtual time function and the packet selection policy. Weighted fair queuing (WFQ) is the best-known PFQ and worst-case fair weighted fair queuing (WF^2Q) is the most accurate PFQ [13]. They use a virtual time function that is defined with respect to the GPS. While this is the most accurate virtual time function, it is too complex to implement since its computation involves keeping track of the number and identity of all the active sessions in a fluid GPS system which may change as many times as there are sessions in the system. The simplification has been with algorithms whose virtual time functions can be calculated directly from the state of the packet system. In Self-clocked fair queuing (SCFQ), the virtual time function is easier to compute but it results in higher delay bounds than WFQ. More efficient time functions are still emerging like in the resulting frame based fair queuing (FBFQ) [13]. In terms of packet selection policy we have the smallest virtual finish time first (SFF) where the packet transmitted is one with smallest virtual finish

time. The algorithms employing this are; WFQ, SCFQ and FBFQ. The other is the smallest virtual start time first (SSF) and is used by SFQ.

WFQ is the most known PFQ algorithm. It overcomes the limitations of FIFO and priority queuing by providing end-to-end guarantees on a per-flow basis. It has been proposed by the IETF [14] as a reference server model for the guaranteed service class in the Internet. WFQ can introduce priorities in the system if it applies weights to the finish numbers and by attaching specific performance bounds to individual flows for data rate by adjusting the flows weight. A flow with higher weight gets lower delay. It provides max-min fairness and the potential for specific QoS bounds. WFQ has its own disadvantages too. It is relatively complex to implement and has a higher packet-processing overhead. It does not scale in number or granularity of classes and it does not ensure explicit rate control for traffic classes. WFQ does not support the bandwidth efficiency of borrowing from among classes. WFQ is normally far ahead of GPS in terms of the number of bits served for a session.

2.3 QUALITY OF SERVICE IN THE INTERNET

Quality of Service can be described as a set of parameters that describe the quality of a specific stream of data. An example of the parameters includes the bandwidth, buffer usage, priority and CPU usage. In the basic IP protocol stack there is no QoS. It employs the best effort scheme. In the best effort scheme packets are transmitted from point to point without any guarantee for special bandwidth or minimum time delay. To provide predictable services in the Internet the IETF developed the Integrated Services (IntServ) and Differentiated Services (DiffSev) and the Multiprotocol Label Swithching (MPLS). The basic idea behind MPLS is to associate labels with IP datagram flows and use these labels to route the datagrams instead of the much longer IP addresses. A summary of the Comparison of differentiated Services and Integrated Services is given in table 2.1.

Table 2.1. A comparison of the IntServ and DiffServ

	Integrated Services	**Differentiated Services**
Granularity of service differentiation	Individual flow	Aggregate of flows
State in routers (e.g. scheduling algorithms, buffer management)	Per- flow	Per-aggregate
Traffic classification basis	Several header fields	The DS field (6 bits) of the IP header
Type of service differentiation	Deterministic or statistical guarantee	Absolute or relative guarantee
Admission control	Required	Required for absolute differentiation only
Signaling protocol	Required (RSVP)	Not required for relative schemes; absolute schemes need semi-static reservations or broker agents
Coordination of service differentiation	End-to-end	Local (per-hop)
Scope of service differentiation	A unicast or multicast path	Anywhere in a network or in specific paths
Scalability	Limited by the number of flows	Limited by the number of classes of service
Network accounting	Based on flow characteristics and QoS requirement	Based on class usage
Network management	Similar to circuit switched networks	Similar to existing IP networks
Interdomain deployment	Multilateral agreements	Bilateral agreements

2.3.1 Integrated Services

The Integrated Services Architecture (IntServ) [14] was defined by the IETF to extend the existing IP architecture to support both real-time and best effort traffic flows. Real-time multimedia applications require QoS guarantees. Video broadcast and audio conferencing software needs a guaranteed bandwidth to provide video and audio in acceptable quality. Due to routing delays and congestion losses, real time applications don't work well on the current best effort Internet. IntServ makes it possible to divide the Internet traffic into standard best effort traffic and application data flows with guaranteed QoS. The IntServ architecture defines a flow as a stream of packets with common source addresses, destination addresses, and port numbers. The architecture suggests that for a flow to receive a desired level of service in terms of quantifiable bandwidth or delay, it is necessary to install and maintain flow specific state in the network. The router in the network exercises control over what flows will be allocated what resources based on the available capacity.

The basic components of the IntServ architecture are traffic control, traffic classes and the setup protocol and its model is as in figure 2.3.

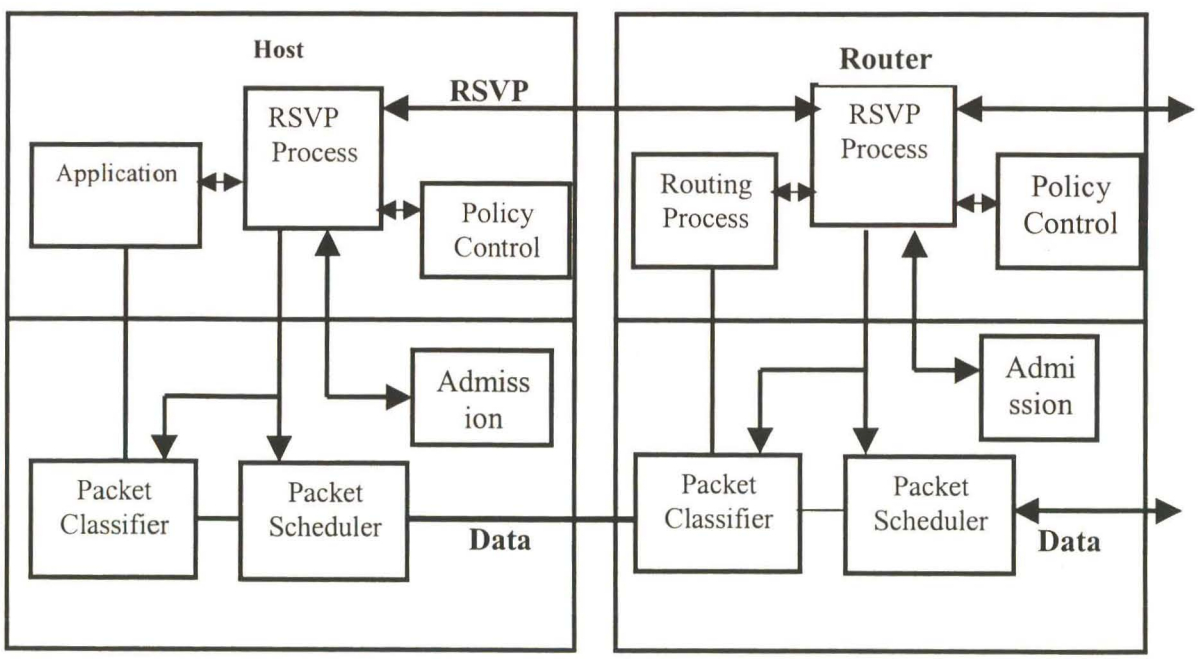


Figure 2.3. The Integrated Services model

2.3.1.1 Traffic Control

The router function that provides different quality of service is called traffic control. The traffic control consists of the admission control, packet classifier and the packet scheduler. The admission control checks if the resources in the router can support a particular QoS for a flow. The packet classifier examines the source address, destination address, and port fields in each packet to determine what class the packet belongs to and hence the type of service it will receive. The packet scheduler schedules the packet for transmission on the outbound link.

2.3.1.2 Traffic classes

IntServ supports guaranteed service, controlled load service and the best effort service classes. The guaranteed service supports real-time traffic flows that require quantifiable bound on delay. The controlled load service approximates best effort service over an uncongested network.

2.3.1.3 Set up Protocol

The setup protocol enables a host or application to request a specific amount of resources from the network. The IntServ architecture uses the Resource Reservation Protocol (RSVP) [16] to set up and control QoS reservations. The basic part of resource reservation is the path. The path is the route a packet takes as it flows through different routers from the sender to the receiver. All packets belonging to a specific flow use the same path. The source sends a path message along the routed path to the unicast or multicast destination. This marks the routed path between the sender and the receiver and also collects information about the QoS viability of each router along that path. The destination receives the path message and gauges what service the network can generate. If the receiver wants to make a reservation for this flow it sends a reservation (Resv) message containing QoS requested from the receiver. At each node the QoS is reserved and the Resv message is send to the next node until the source. The per-flow reservation state maintained in the router will be deleted unless RSVP Path and Resv messages are periodically sent by the senders and receivers, respectively.

2.3.1.4 Limitations of the Integrated Services

- Scalability

  IntServ has a limitation with regards to its applicability and scalability over large networks. If the network is big, IntServ is required to maintain state information on so many flows in the network.

- Security

  The current version of RSVP lacks both adequate security mechanisms to prevent unauthorized parties from instigating theft of service attacks, and policy control, i.e. techniques to authenticate and authorize applications or end users wishing to reserve resources.

- Signaling

  The need for a signaling protocol is also a limitation. Signaling is required in connection oriented networks, but datagram networks are connectionless and typically don't require signaling.

2.3.2 Differentiated Services

Differentiated Services (DiffServ) [17] is the other approach for supporting IP QoS. It has gained popularity due to the following reasons. Firstly, it is scalable since individual host-to-host microflows are aggregated into a single larger aggregate flow and then that single flow receives special treatment. Secondly it is applicable to all applications and doesn't require special control protocols like the RSVP. It does not burden the router and switches with per flow or per customer state information. It provides different quality of services and can be used for charging customers. The approach taken by DiffServ is to classify individual microflows at the edge of the network into one or several unique service classes and then apply per-class service in the middle of the network. The classification is done at the network ingress based on an analysis of one or more fields in the packet. The packet is then tagged (turning some bits in the packet header) as belonging to a particular class and then injected into the network. The core routers that forward the packet examine the tag to provide different kinds of service discrimination.

2.3.2.1 DiffServ Architecture

Unlike IntServ, QoS guarantees with DiffServ are static and stay long term in routers. There are no QoS reservations. The data needs to be marked by differentiated services (DS) byte which is interpreted by routers in the network.

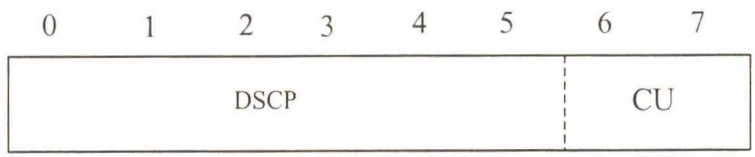


Figure 2.4. Differentiated Services field in the IP Header

To accomplish the provision of different service classes, Diffserv uses the following architecture.

- Per-hop-Behavior

The Type-of-service (TOS) field in Ipv4 and the traffic class field in Ipv6 have been redefined as the differentiated services (DS) fields. This is as shown in figure 2.4. Six bits of the DS fields are used as the differentiated service code point (DSCP) to select the traffic class a packet experiences at each node. The other two bits currently unused (CU) are reserved and can be used latter. Each DS capable network should have information on how packets with different DS fields should be handled. In DiffServ specification this information is called the Per-Hop-Behavior (PHB). It is a description of the forwarding treatment a packet receives. The DSCP value in the DS field is used to select the PHB a packet experiences at each node. To provide predictable services, PHB need to be available in all routers in a DiffServ capable network. The PHB can be described as a set of parameters inside a router that can be used to control how packets are scheduled onto an output interface. DiffServ requires routers that can support queue scheduling and management to prioritize outbound packets and control the queue length to minimize congestion on the network like the RED algorithm of Section 2.2.3. A group of packets with the same DSCP is called a behavior aggregate (BA). A PHB is applied to each BA inside a network.

- Differentiated Services Domain

The IETF working group defines a differentiated services domain as a contiguous portion of the Internet over which a consistent set of DiffServ policies are administered in a

coordinated fashion. A DiffServ domain consists of boundary components that are used to connect different DiffServ domains to each other and interior components that are only used inside the domains. It normally consists of one or more networks under the same administration. The administration of a DiffServ domain is responsible for ensuring that adequate resources are provisioned and provided to support the service level agreements (SLA's) offered by the domain.

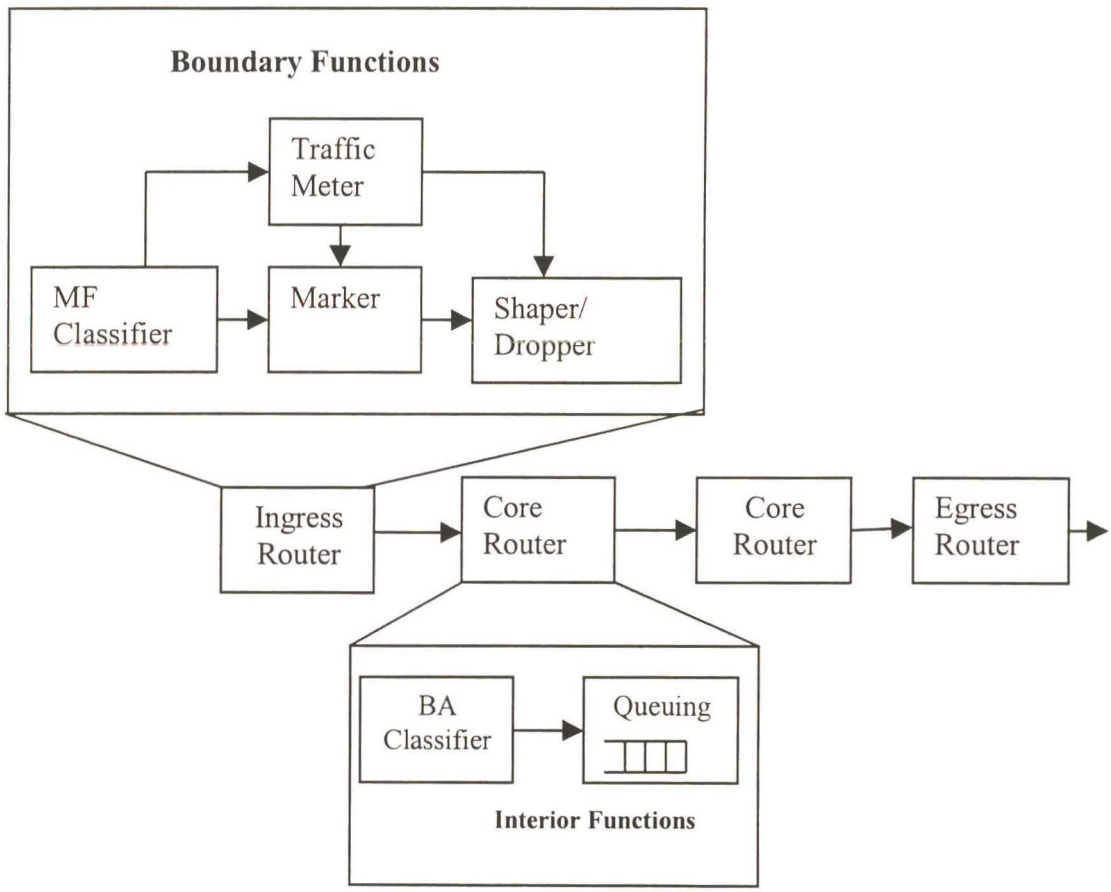


Figure 2.5. The DiffServ Boundary and Interior Elements

- DS Boundary nodes

All data packets that travel from one DiffServ domain to another must pass a boundary node which can be a router, a host or a firewall. A DS boundary node that handles traffic leaving a DS domain is called an egress node whereas the one that handles traffic entering a DS domain is called an ingress node. Normally a DS boundary node acts as both ingress node and egress node depending on the traffic direction. The ingress node

ensures that the packets entering the domain must receive the same QoS as the domain the packets traveled before. A DS egress node performs conditioning functions on traffic that is forwarded to a directly connected peering domain. The boundary nodes consist of a traffic conditioner which classifies, marks and possibly conditions packets that go through it. It consists of several components as discussed below.

- Classifier: The classifier selects packets based on their packet header and forwards the packets that match the classifier rules for further processing. The DiffServ model specifies two types of classifiers, a multifield classifier which can classify on a DS field as well as any other IP header field and a BA classifier which classify only on the bits in the DS field.
- Meter: Traffic meters measure if the packets selected by the classifier corresponds to the traffic profile that describes the QoS for the SLA between the customer and the service provider. A meter passes state information to other conditioning functions to trigger a particular action for each packet which either does or does not comply with the required QoS requirements.
- Marker: DS markers set the DS (PHB) field of the incoming IP packet to a particular bit pattern.
- Shaper/Dropper: Packet shapers and droppers cause conformance to some configured traffic properties, e.g. a token bucket filter. They use different methods to bring the stream into compliance with a traffic profile. A shaper delays some or all the packets. It has a finite buffer and can drop some packets if it has no space. A dropper discards some or all the packets. This process is known as policing the stream. A dropper can be implemented as a shaper with buffer size of zero.

- DS Interior Components

DS interior components can be core switches or routers that provide the PHB based on the DSCP bits contained in the DS field. These devices typically employ a queue management and scheduling discipline to provide the PHB. An example of the mechanisms used is RED and WFQ.

2.3.2.2 DiffServ Classes

There are several proposed models of DiffServ classes, most of which have not been standardized. The DiffServ models fall into two categories the assured service scheme proposed in [2] and the premium service scheme proposed in [3]. They add on the traditional best effort traffic class. The other models combine in some way or another the principles used in the assured and premium schemes. Some are based on more sophisticated scheduling mechanisms, like the user share differentiation (USD) scheme [18] which is based on the principle of link sharing.

2.3.2.2.1 Assured Service

Clark [2] argued that a guaranteed, or at least expected throughput is one of the QoS features of interest to most applications. The impact on the user of using congested network is not a constant increased delay to individual packets but a reduction in throughput for data transfers. He argues that packet delay is not an indication of service quality. The assured service model [19] is based on this argument. It does not provide for absolute guarantees but assures the user of available bandwidth along certain network paths. This model is based on some provisioning algorithm and route determination. The application requesting assured service must know the size of the data to be transferred and the desired delivery time. This can be converted into a minimum transfer rate which becomes the service objective of the transfer. The rate then becomes the service profile for the transfer, and is installed in the traffic meter at the source. The meter monitors the transfer in progress and tags each packet of the data stream as whether it is IN or OUT of the service profile. There is an IN/OUT flag that is set in the packet depending on whether it is IN or OUT. Implementing the profile meter depends on how the problem has been defined. For example a profile that specifies a mean rate and a maximum burst length can be implemented using a leaky bucket filter. The leaky bucket filter (token bucket filter) works as follows. It is a bucket of depth B that is replenished with tokens at a rate of R tokens per second. When a packet arrives at a router some number of tokens are subtracted from the bucket. A traffic source is said to conform to the parameters of the token bucket filter if it sends packets at a rate less than or equal to R. The

conforming traffic never exceeds $R(t)+B$ for any increment of time t. Conforming traffic is marked as IN while nonconforming traffic is marked as OUT.

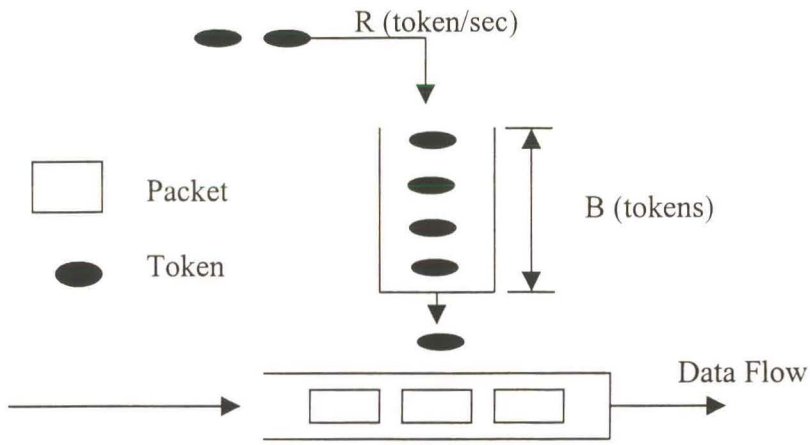


Figure 2.6 Token bucket filter

Another type of profile meter is the Time Sliding Window [20]. It has a rate estimator and a tagging algorithm. It provides a smooth estimate of the TCP sending rate over a period of time. With the estimated rate the tagging algorithm can tag packets as OUT once the traffic exceeds a certain threshold. In case of congestion the tag information is used by intermediate routers in the network to discriminate against OUT traffic by preferentially dropping them. The selective drop mechanism can be implemented using a threshold mechanism, pushout mechanism and RED. In the threshold mechanism when the buffer occupancy reaches a given threshold, arriving OUT packets are discarded while IN packets are still admitted as long as the buffer is not full. The threshold mechanism can be extended to the pushout mechanism where when the buffer is full OUT packets can be dropped to give room for IN packets. However the most widely used selective dropping scheme is the RED algorithm with IN and OUT (RIO) mechanism [20]. RIO uses the same mechanism as the RED algorithm but uses two sets of parameters, one for IN packets and the other for OUT packets. When a packet arrives at the router, it checks whether the packet is IN or OUT. If the packet is IN the router calculates the average queue for the IN packets $avg-in$ and then calculates the probability of dropping the packet from this value. If it is an OUT packet the router calculates the average total queue size for both IN and OUT arriving packets $avg-total$

and then calculates the probability of dropping the packet from this value. As in Section 2.2.3 the RED has the following parameters; the minimum threshold, the maximum threshold and the maximum probability, $\min-in_{th}$, $\max-in_{th}$, $P_{\max-in}$ for the IN packets and $\min-out_{th}$, $\max-out_{th}$, $P_{\max-out}$ for the OUT packets.

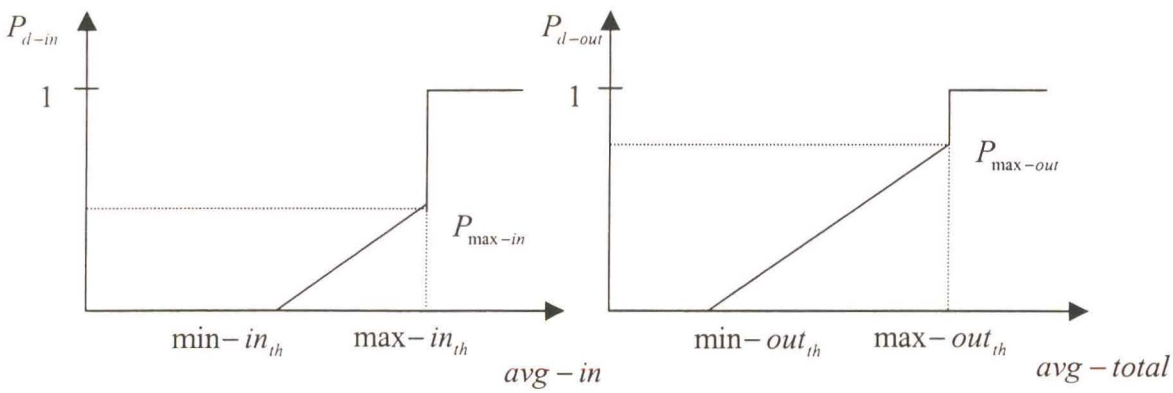


Figure 2.7. RIO algorithms (Figures not drawn to scale)

Discrimination against out packets is achieved by the choice of the RED parameters as illustrated in figure 2.7. This is achieved in the following ways.

- Setting $\min-in_{th} > \min-out_{th}$ and thus the OUT packets are dropped earlier than IN packets.
- Setting $P_{\max-out} > P_{\max-in}$ and thus dropping the OUT packets with a higher probability than the IN packets.
- Setting $\max-in_{th} > \max-out_{th}$ and thus it starts dropping all IN packets earlier than OUT packets.

For the choice of average queue values, we use $avg-in$ for IN packets since the network can provision for the amount of IN traffic. The OUT traffic represents opportunistic traffic and there is no indication of the proper amount of traffic. Therefore we use $avg-total$ for OUT packets and not $avg-out$ as would be expected since these would not cover the conditions when the queue is growing due to arriving IN packets. The IN packets have a low loss rate even in congestion and therefore get a higher throughput. The end user whose traffic is conformant to a certain profile and the packets are tagged IN will perceive a predictable service. The assured service is built with a

profile meter (in combination with the mechanisms to determine the traffic profile) and buffer management schemes in the routers. Assured service could be the service that an (Internet service provider) ISP would provide to individual customers who are willing to pay a bit more for Internet services that seem to be unaffected by congestive periods.

2.3.2.2.2 Premium Service

Premium service class [3] is for delay sensitive applications and provides a "virtual leased line" [21]. The applications are specified with the desired peak rate and burst size. The assurance is that the application should not exceed the peak rate and the network guarantees a contracted bandwidth. The edge devices filter and tag the packets entering the network and perform traffic shaping on the flow hence smoothing all traffic bursts before they enter the network. Two levels of priority queuing are implemented. Tagged packets are sent first while non-tagged packets (best effort) are queued and sent later. Premium service can be looked upon as consisting of two networks.

- Firstly, premium traffic consists of a virtual network where traffic is limited and shaped to a contracted peak-rate. Here the packets move through a network of queues where they experience almost no queuing delay. The traffic patterns are regular, and they have small or nonexistent queues.
- Secondly, it consists of the normal best effort IP network where the traffic, which is bursty, requires queue management to deal fairly with congestive periods.

The service is implemented by admission control and policing at edge routers and then, in the forwarding paths, the actions are to classify a packet into one of two queues based on the priority bits and to service the two queues using simple priority. The admission control shaping must include both rate and burst parameters. A simple token bucket implements the policing (shaping). Premium service is to be priced higher than the best effort traffic since it has some form of resource reservation. Premium service scheme has strong appeal for commercial applications, video broadcasts, voice over IP and VPN's.

2.3.2.2.3 Combined Model

A good example of a model which combines both the assured and premium traffic is the two bit differentiated services architecture as proposed by Nichols et al. [3]. Two bits from the IP header precedence field can be designated for premium traffic (P-bit) and the other for assured traffic (A-bit). The packet forwarding mechanisms can be broken into those at the input interface and those at the output interface. Intermediate routers only need to implement the post packet forwarding functions, while leaf and border routers must perform functions on arriving packets before forwarding. Leaf routers are configured with a traffic profile for a particular flow based on its packet header. All arriving packets have their A and P bits cleared after which they are passed to a classifier where they are classified on their header. From the classifier, the classified packets are then forwarded to a marker. The markers have been configured for the usage profile of the flow. They use the token packet principle as in figure 2.8 and it fills at the flow rate specified by the usage profile. When the token is present, assured packets have their A-bit set, otherwise the packets are passed to the forwarding engine unmarked. For premium service in case there is no token they are held until a token is present then set their P-bit and passed to the forwarding engine.

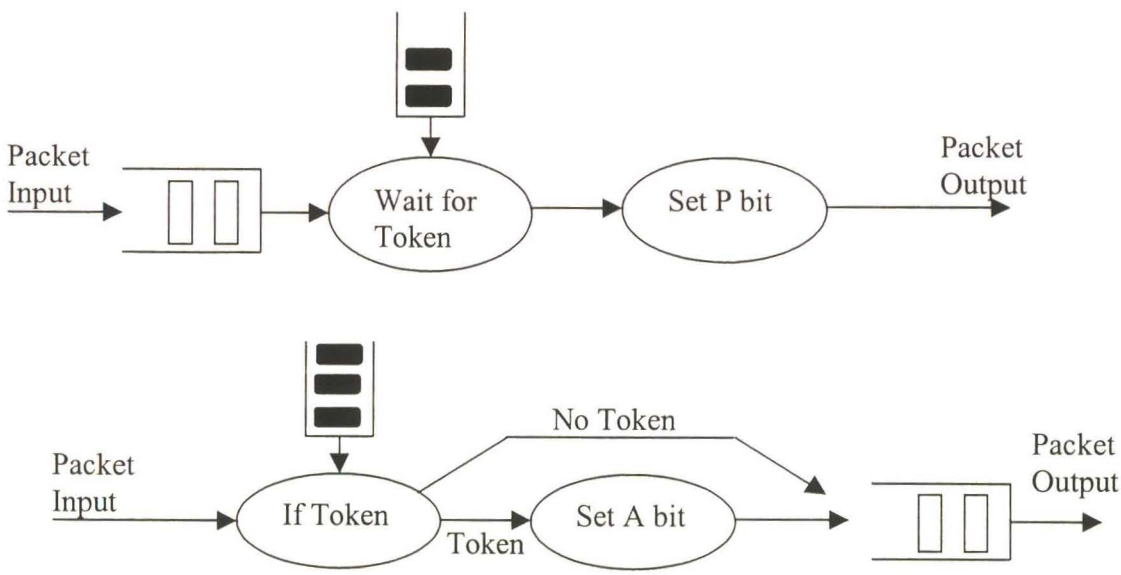


Figure 2.8 Markers for two different services

The output interface of a router has two queues; the high priority queue for the premium packets and the low priority queue for both assured and best effort (unmarked) packets. The low priority queue implements the RIO algorithm as discussed where the unmarked packets are the OUT packets. Figure 2.9 shows a block diagram of the output interface for all routers.

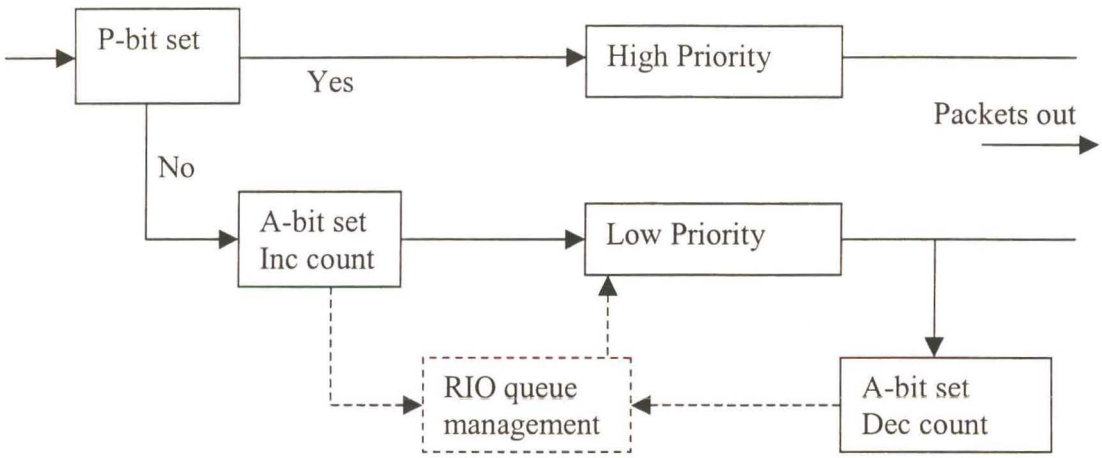


Figure 2.9. Router Output Interface

The premium service is burst and bandwidth controlled and thus cannot starve best effort traffic. Assured service bursts can however starve best effort traffic during periods of congestion. The border router has a profile meter whose token bucket filter is configured with the contracted rate across the link. The intermediate routers employ the mechanisms (per-hop-behaviors) of any differentiated services network. This combined model is a very realistic model for the network since the network contains a mixture of many traffic types which are compatible in the network. The premium service implements a guaranteed peak bandwidth service with negligible delay and cannot starve best effort and its allocation is straightforward. Assured traffic permit traffic flows to use excess bandwidth and might be a service fit for ISP. If the services are deployed together the unused premium bandwidth can be used by the assured service.

2.4 RESULTS

2.4.1 Assured Simulation Results

A simulation was conducted to show throughput of an assured buffer with a varying amount of IN packets. In our simulation model we assume a buffer of size B (100 in the simulations). The packets arrive according to a Poisson distribution with rate of λ. The packets are marked IN with a probability p_{in} and out with a probability $p_{out} = 1 - p_{in}$. The packets require a service that is exponentially distributed with parameter μ (100packets per second in the simulations). The total offered load to the system is $\rho = \lambda/\mu$. The buffer employs the RIO mechanism with parameters chosen as follows: $\min\text{-}in_{th} = \min\text{-}out_{th} = B/2$, $\max\text{-}in_{th} = \max\text{-}out_{th} = B$. The marking probabilities are as follows, $P_{\max\text{-}in} = 0.1$ and $P_{\max\text{-}out} = 0.9$. We simulate four sources with different marking probabilities: $prob1 = 0.9$, $prob2 = 0.7$, $prob3 = 0.5$ and $prob4 = 0.3$. We obtain the results as shown in figure 2.10.

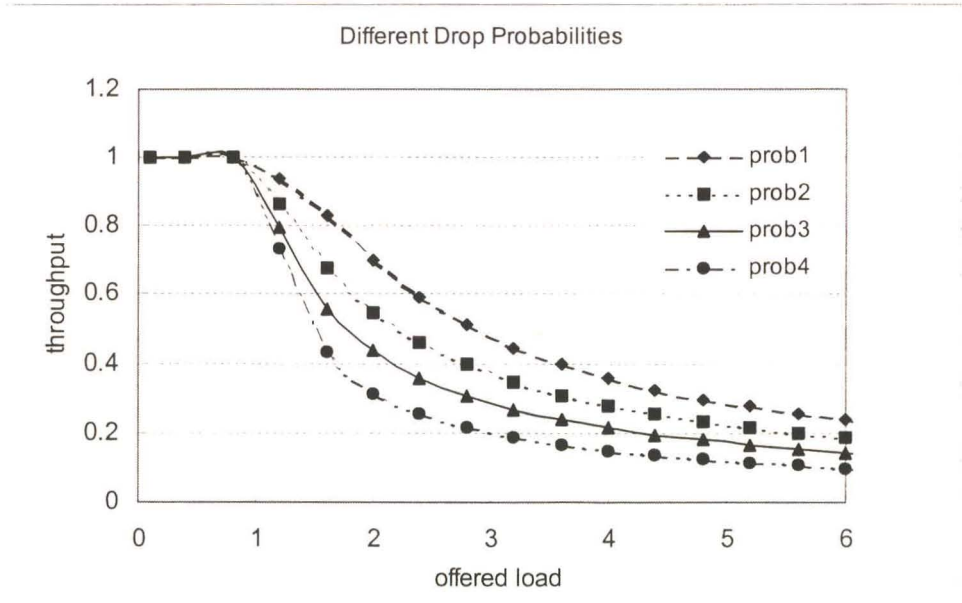


Figure 2.10. Assured Simulations

From the results we can clearly see that as we increase the marking probability the throughput increases. This can be explained by the fact that at a higher probability we

have more IN packets than at a lower probability and the IN packets have lower probability of being lost than the OUT packets. Also we notice that the throughput reduces with an increase in the offered load. This is because as more packets arrive the queue becomes full and therefore the RIO algorithms become more aggressive in dropping packets.

The RIO (RED) algorithm is not the only mechanism that can be used to discriminate against packets in the network. The others as mentioned are the threshold dropping scheme and the drop tail scheme. We compare how the different packets share the bandwidth with these various schemes. For the threshold dropping, IN packets are accepted in the queue as long as it is not full, while the OUT packets are accepted as long as the queue is less than half full. For the drop tail scheme all the packets are accepted in the queue as long as it is not full. The packet marking probability was taken to be 0.5. The results are plotted in figure 2.11

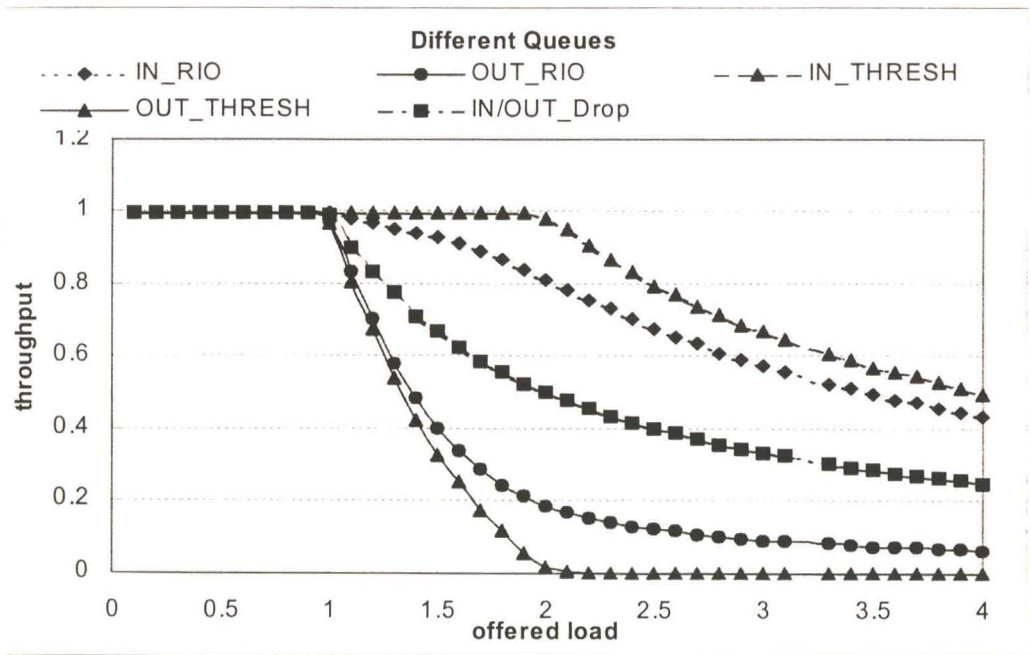


Figure 2.11. Different Queue Mechanisms

The results in figure 2.11 show that with the tail drop scheme the throughput for both IN and OUT packets are the same. When the offered load $\rho < 1$, the throughput doesn't

depend much on the queue scheme. However, when $\rho > 1$, for RIO and threshold, there is discrimination between the IN and the OUT packets, with the IN packet receiving a higher throughput than the OUT packets. Their values converge as the offered load increases.

2.4.2 Simulation Results for the Combined Model

A simulation was conducted to show the effect of having both assured and premium traffic on the network. This was proposed in [3] but there are no results supporting the proposal. The combined simulation model is as in figure 2.12. The premium packets arrive with a Poisson rate of λ^P and are marked with a probability $p^P(0.9)$ as being IN and $1-p^P$ as being OUT. The Premium IN packets at rate λ_{in}^P are forwarded into the higher priority queue while the OUT packets at rate λ_{out}^P are forwarded into the low priority queue where they are treated as best effort traffic. The assured packets arrive with a Poisson rate of λ^A and are marked with a probability $p^A(0.7)$ as being IN and $1-p^A$ as being OUT. The assured IN packets at rate λ_{in}^A are forwarded into the lower priority queue while the OUT packets at rate λ_{out}^A are forwarded into the low priority queue where they are treated as best effort traffic.

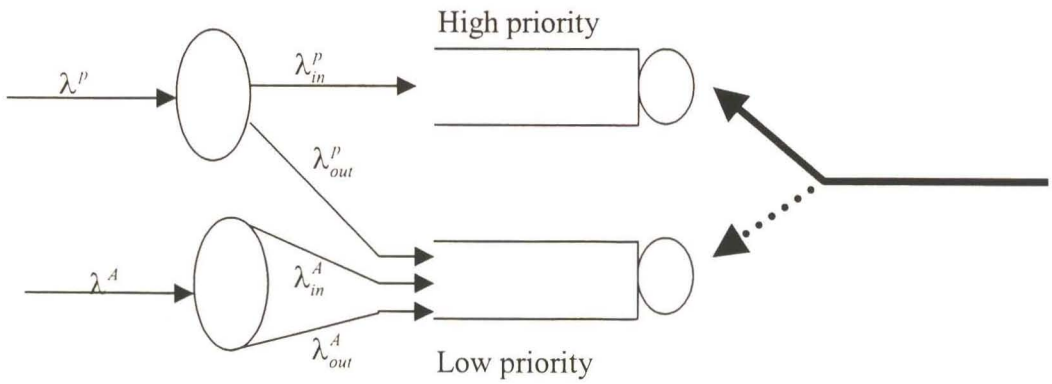


Figure 2.12 Combined model

The low priority queue employs the RIO algorithm. The service discipline for both queues is a non pre-emptive priority discipline. The buffer size for both the high and low priority buffers was taken to be 100. The results of the throughput vs. the offered load

and packet loss probability vs. the offered load are plotted in figure 2.13a and 2.13b respectively.

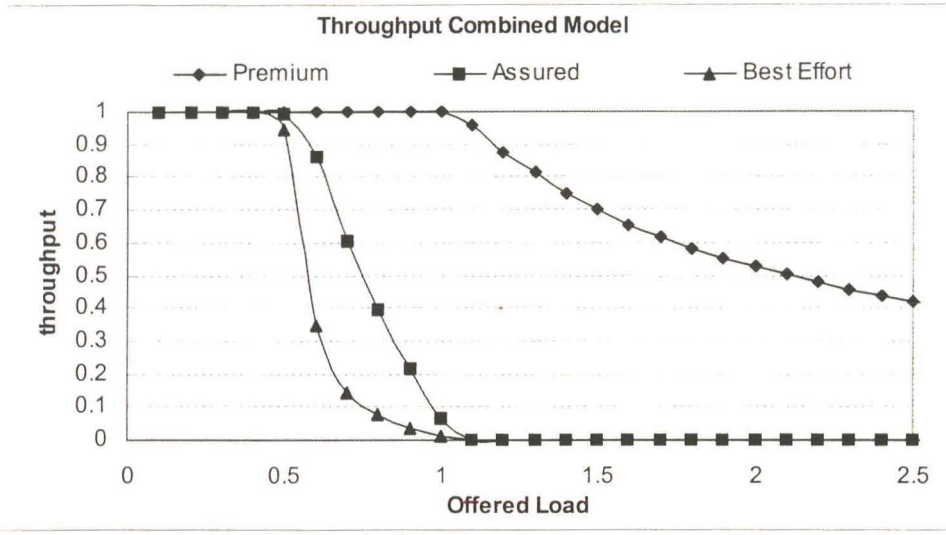


Figure 2.13 a) Throughput

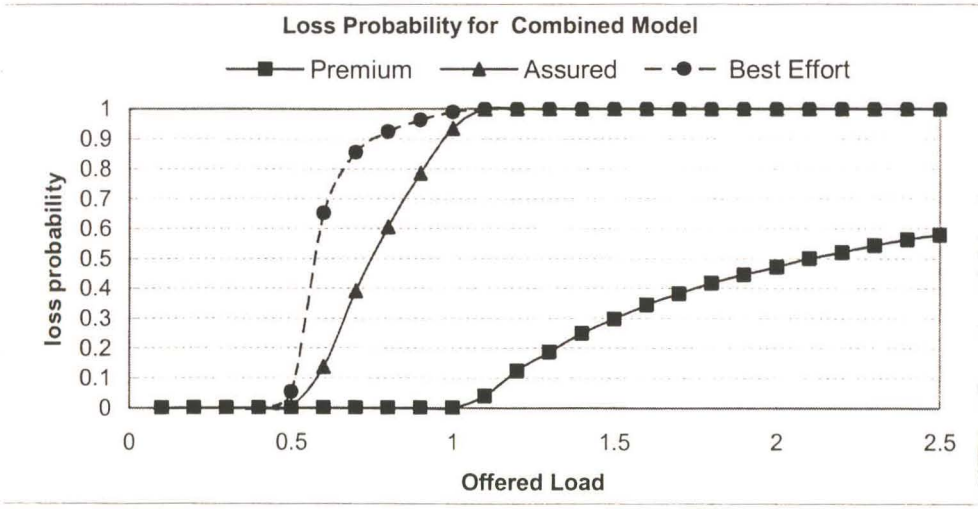


Figure 2.13 b) Loss probability

Figure 2.13. Combined model results

From the results it can be seen that the throughput for all classes of traffic reduces as the offered load increases. This is due to the higher number of packets which leads to higher drop probabilities as indicated by the drop probabilities graphs. The reduction rate is

however different for the classes. At $\rho = 0.5$, the throughput of the best effort traffic starts reducing tremendously followed by the assured traffic. This is because there is a significant amount of premium traffic in the high priority buffer and hence the low priority buffer is rarely served. In fact the throughput goes to zero as the drop probability increases to one. This factor of the throughput of the best effort traffic going to zero with an increase in premium traffic leads to the strict conditioning of premium traffic. If loosely conditioned the premium traffic can starve the other traffic in the network.

Premium traffic is suitable for traffic with minimum delay guarantees. To see this we simulate the delays seen by the traffic in our simulation model of figure 2.12 as described before. The results are plotted as in figure 2.14. From the figure the premium traffic has negligible delay as compared to the assured and best effort traffic. Hence, it can be used for the delay-constrained traffic.

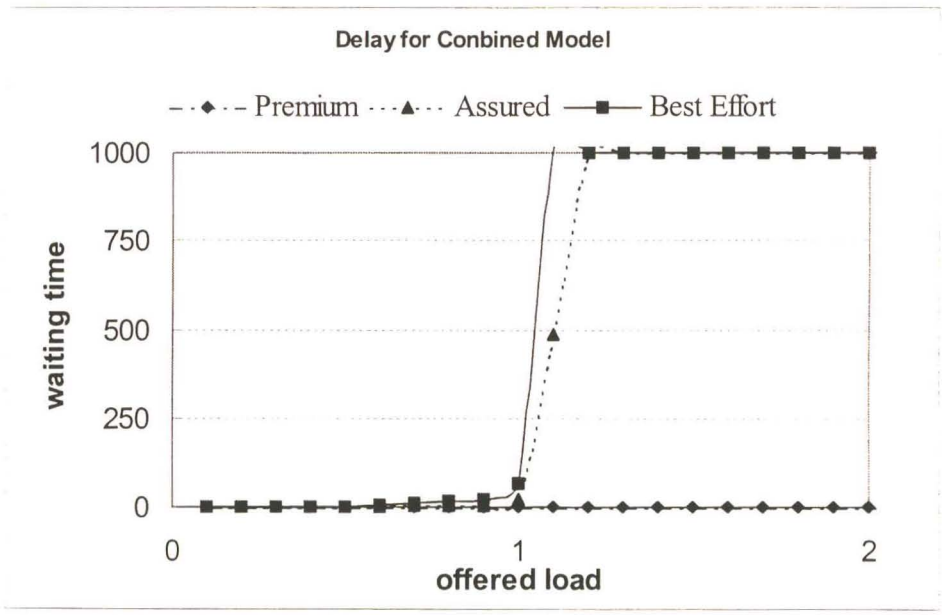


Figure 2.14. Delay guarantees

2.5 RELATIVE AND ABSOLUTE DIFFERENTIATED SERVICES

The absolute service differentiation is a way of meeting some of the goals of absolute performance levels like the IntServ but without per-flow-state in the backbone routers, and with only semi-static resource reservations instead of a dynamic resource reservation

protocol. In this scheme the user is assured his requested performance level. Its disadvantage is that the user will be rejected if the required resources are not available and the network cannot provide the requested assurances. For example, suppose that an IP telephony application requests a bandwidth of 32kb/s and an end-to-end delay at most 200ms. If the users request is accepted, the quality of the ensuring call is assured. However, if the network is unable to provide the requested bandwidth and/or end-to-end delay the user will receive a busy signal.

The relative service differentiation [22] involves service models that provide assurances for the relative quality ordering between classes, rather than for actual service level in a class. The central premise in relative differentiated services scheme is that the network traffic is grouped into N service classes, which are ordered based on their packet forwarding quality. Class i is better (or at least no worse) than class $(i-1)$ for $1 < i \leq N$, in terms of local (per-hop) performance measures for queuing delays and packet losses. The only assurance from the network is that a higher class will receive better service than a lower class. The amount of service received by a class and the resulting QoS perceived by an application depend on the current network load in each class. A relative Service differentiation model should have the following key features:

- Controllability: the network operators should be able to adjust the quality spacing between classes based on their pricing policy criteria.
- Predictability: class differentiation should be constant (i.e. higher classes are better or at least no worse) even in short time scales, independent of the variation in class loads.

2.6 SUMMARY

The IETF has proposed the Diffserv and IntServ for providing QoS. The router and therefore the IP layer is the best place for enforcing QoS parameters. This is because the Internet router is the best candidate for detecting congestion and taking appropriate action like dropping the packet. It can distinguish between propagation delay and persistent delay furthermore it has a unified view of the queuing behavior over time and it is also

shared by a wide range of traffic. The router enforces QoS with the help of management of the router buffers and scheduling mechanisms. Scheduling algorithms are for resource management in the network. They need support from buffer management strategies since the buffer size is limited and the packets would eventually be dropped. In this chapter we have looked at the various buffer management mechanisms and scheduling mechanisms that are used to provide for different QoS. We have particularly focused on the RED which is currently the supreme buffer mechanism for providing for assured traffic. We further look at the assured service scheme with IN and OUT packets. Through simulation we have found that different traffic with different number of IN and OUT packets when passed through a RIO buffer achieve different throughput. Therefore DiffServ is possible with RIO. We have further compared RIO with threshold dropping and found RIO performing better than threshold dropping. We have also looked at a network with premium and assured traffic. We find that the premium traffic undergoes no delay as compared to assured and best effort. Therefore premium traffic can be used for delay sensitive applications.

CHAPTER 3
TRANSPORT CONTROL PROTOCOL AND ITS PERFORMANCE

3.1 INTRODUCTION

Due to the popularity of TCP we need to understand the performance of the transmission protocol and how its parameters such as the timeout time, the loss window and others respond to external network conditions like the retransmission time and the packet loss. In the literature there have been several papers that have considered the performance of the transport control protocol. Most of these papers are based on simulations [24][25]. There are very few papers that have presented the analytical model [26][27][28]. These models have a lot of limitations in analyzing the many parameters of TCP. In this chapter we develop an analytical model for the transport control protocol. Our analytical model is based on the packet train model of [33] with several modifications. We stochastically look at the TCP window evolution since it's the key to TCP performance. The analytical model is adaptive and will be used to incorporate many parameters in the following chapters of the thesis. We use our analytical model to study the way the transport protocol responds to some issues like an increase in the TCP packet loss, the variations of

the time various packets travel from the transmitter to the receiver and the time the acknowledgement is received by the transmitter (round trip time), the effect of the limitation of the TCP maximum window (the window is limited due to the finite receiver buffer) and the effect of the TCP timeout value on the throughput performance of TCP. The analytical results are further compared with simulation results obtained from a discrete event simulator. We also compare them with the formulas obtained from [27] and [28] where applicable. These formulas have not taken into account all the aspects (parameters) of the transport protocol and are further limited in their scope of application.

3.2 ALGORITHMS FOR VARIOUS TCP PROTOCOLS

A complete TCP Protocol is described in [29]. However we give the basic algorithms necessary for the performance analysis of TCP. The TCP Protocol has both the sending and receiving side maintaining windows. The receiver, which has a finite resequencing buffer, accepts out of sequence packets and delivers them in sequence. It advertises a maximum window size at connection set up of which the transmitter window cannot grow beyond. The receiver returns an ACK for every good packet it receives. An ACK packet that acknowledges the first receipt of an error-free-in-sequence packet is called the first ACK. The ACK's are cumulative; hence they carry the next expected packet number. The TCP sender maintains the TCP window. It also maintains several variables at all time t for each connection and they are presented as below [30].

- It maintains a lower window edge $A(t)$ below which all packets have been acknowledged. $A(t)$ is non-decreasing. The receipt of an ACK with sequence number $n > A(t)$ causes $A(t)$ to jump to n.
- It maintains the congestion window $W(t)$. The transmitter can send packets with sequence numbers n, such that $A(t) \le n < A(t)+W(t)$. It should be noted that $W(t) \le W_{\max}$.
- It maintains the slow start threshold $W_{th}(t)$. This controls the increments in $W(t)$ as explained below.

The basic TCP window adaptation algorithm is as in [30]. The normal window evolution is triggered by the first ACK's and timeouts as follows.

- During slow start, when $W(t) < W_{th}(t)$, each first ACK causes $W(t)$ to be incremented by one.
- During congestion avoidance, when $W(t) \geq W_{th}(t)$, each first ACK causes $W(t)$ to be incremented by $1/W(t)$. Therefore the window increases by one if all the ACK's of the window have been received.
- If timeout occurs at time t, and t^+ is a short time after a timeout has occurred, $W(t^+)$ is set to 1, $W_{th}(t^+)$ is set to $W(t)/2$, and retransmission begins from $A(t)$.

The transmitter performs several tasks, some of them are outlined as below.

- The transmitter runs the basic window adaptation algorithm described above.
- The transmitter runs the loss detection mechanism. This is a mechanism by which the transmitter concludes (correctly or incorrectly) that a packet was lost. A packet loss is detected by a timeout or the reception of a certain number of duplicate ACK's, a process called first retransmit.
- The transmitter runs the loss recovery. This is a mechanism which allows the protocol to recover lost packets through retransmissions.
- The transmitter runs the window adaptation algorithm during loss recovery.

Different TCP versions differ in the way they detect and recover from losses. Some of the versions of TCP are discussed below.

- For Old Tahoe loss detection and recovery are performed only through timeout and retransmission where the transmitter waits for a timeout to recover from a packet loss. Window adaptation during loss recovery follows the basic algorithm.
- Tahoe uses both timeout and fast retransmit for loss detection after which it behaves as if a timeout has occurred.
- Reno uses fast retransmit for loss detection. It employs fast but conservative recovery where it remains in the congestion avoidance phase after a packet loss. It can recover effectively one lost packet per window.

- New Reno uses fast retransmit to detect losses and the fast recovery to recover from packet losses. The protocol sends a lost packet with the receipt of the first ACK unlike Reno, which needs a certain threshold of packets to send the next lost packet. It actually recovers one lost packet per retransmission timeout interval.
- The selective acknowledgement (SACK) is an extension of New Reno. However, each ACK received by the sender contains information about any non-contiguous set of data that has been received and queued at the receiver, hence the sender infers different packet loses and can send multiple lost packets per round trip time. The more recent method of interest is the Duplicate SACK extension to TCP. This allows the data receiver to report the receipt of duplicate segments. Hence the sender can infer whether or not the retransmission was necessary. Relevant action is taken after that. One of the proposed actions can be for the sender to undo the halving of the congestion window and setting the values to the old ones.
- The previous TCP versions estimate the available bandwidth in the network based on packet losses, after which they adjust their transmission windows. This congestion avoidance mechanism causes periodic oscillations in the window size due to constant update of window size. This result in larger delay jitter and inefficient use of the available bandwidth. A different version of TCP called TCP Vegas has been developed. It uses a different bandwidth estimation scheme. It uses the difference between the expected and actual flow rates to estimate the available bandwidth and adjusts its window accordingly as indicated in [31].

3.3 PERFORMANCE OF THE TRANSMISSION CONTROL PROTOCOL

3.3.1 Existing Models

Research has been done in the performance of TCP protocols over wired and wireless networks. Most of the research models don't take into account the various parameters that govern the performance of TCP protocols. As seen in Section 3.2, the transmission control protocol has various parameters that affect and govern its performance. The throughput of TCP depends mainly on the following parameters: the round trip time t_{Rtt}, the retransmission timeout value t_{Rto}, the segment size s and the packet loss rate p. The

basic model that approximates TCP throughput T is given by equation (3.1) [27][32]. This model is a simplification and does not take into account TCP timeouts. It is only effective when the loss $p << 1$. The other relatively better model that takes into account timeouts [28] is given by equation (3.2). Both the models work for TCP Reno only.

$$T(t_{Rtt}, s, p) = \frac{c.s}{t_{Rtt}.\sqrt{p}} \tag{3.1}$$

$$T(t_{Rtt}, t_{Rto}, s, p) = \begin{cases} \dfrac{\dfrac{1-p}{p} + \dfrac{W(p)}{2} + Q(p, W(p))}{t_{Rtt}(W(p)+1) + \dfrac{Q(p, W(p))G(p)t_{Rto}}{1-p}} & \text{if } W(p) < W_m \\ \dfrac{\dfrac{1-p}{p} + \dfrac{W_m}{2} + Q(p, W_m)}{t_{Rtt}\left(\dfrac{W_m}{4} + \dfrac{1-p}{pW_m} + 2\right) + \dfrac{Q(p, W_m)G(p)t_{Rto}}{1-p}} & \text{otherwise} \end{cases} \tag{3.2}$$

where W_m is the maximum window size and $W(p)$, $Q(p,w)$ and $G(p)$ are given by

$$W(p) = \frac{2}{3} + \sqrt{\frac{4(1-p)}{3p} + \frac{4}{9}} \tag{3.3}$$

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

$$G(p) = 1 + p + 2p^2 + 4p^3 + 8p^4 + 16p^5 + 32p^6$$

In [26] they derive the TCP utilization ρ_c for TCP Reno given the packet rate μ, and buffering at the bottleneck node, b. In its most simplified form the equation is given by

$$\rho_c = \frac{\lambda_c}{\mu} \tag{3.4}$$

where λ_c is given below

$$\lambda_c = \begin{cases} \dfrac{3\mu(b+t\mu)^2}{4(b^2 + bt\mu + (t\mu)^2)} & \text{if } b < \mu t \\ \mu & \text{otherwise} \end{cases} \tag{3.5}$$

3.3.2 TCP Window Evolution Model

The train model as used in [33] is growing in popularity in analyzing the TCP window. In the basic framework of the train model the TCP cycle consists of two phases; phase one is the slow start or congestion avoidance phase and phase two is the fast recovery or timeout. Each phase is further modeled into rounds or trains. A round starts with the transmission of W packets where W is the current size of the congestion window. Hence the first round corresponds to the first window of packets. Different models now differ on the way they handle the rounds and the analysis of various parameters. In our model we classify the rounds into type one for phase one and type two for phase two of the cycle. In our analysis we use the following notation. Let x be the number of rounds in a cycle and x_1 be the number of type one rounds, x_2 be the number of type two rounds. Let σ be the loss window size, η be a window size which can also be written as η_i to emphasize that it is in round i. Let W_m be the maximum window size, P_t be the steady state probability of a timeout, Q_c be the number of packets transferred during a cycle, Q_1 be the packets sent in phase one, Q_2 be packets send in phase two and Q_r be the packets send after a packet loss and before it is detected by the sender. The total packets transferred is given by

$$Q_c = Q_1 + Q_2 + Q_r \tag{3.6}$$

If we consider a link of higher capacity and assume instantaneous acknowledgements we note that Q_r is small and can be neglected. The average of Q_c is found by

$$E(Q_c) = E[E(Q_c/x_1, x_2, \sigma)] \tag{3.7}$$

$$= \sum_{\sigma}\sum_{x_1}\sum_{x_2} P(x_1, x_2, \sigma)E(Q_c/x_1, x_2, \sigma)$$

$$E(Q_c) = \sum_{\sigma} P(\sigma)\sum_{x_1} P(x_1/\sigma)\sum_{x_2} P(x_2/x_1, \sigma)E(Q_c/x_1, x_2, \sigma)$$

and $E(Q_c/x_1, x_2, \sigma) = x_2 + E(Q_1/x_1, \sigma)$ where we assume that in the loss recovery phase a packet is sent in one round. Therefore we have our equation of the actual number of packets transmitted given by

$$E(Q_c) = \sum_{\sigma} P(\sigma)\sum_{x_1} P(x_1/\sigma)\sum_{x_2} P(x_2/x_1, \sigma)[x_2 + E(Q_1/x_1, x_2, \sigma)] \tag{3.8}$$

Let r_k be the round trip time taken by a packet of the k th phase one round, τ_t be the timeout interval value. Then we can assume that $E(r_k) = R_{tt}$, where R_{tt} is the average round trip time of the network. The total cycle transmission time T_c can be given by

$$T_c = \sum_{k=1}^{x_1+1} r_k + x_2 R_{tt} + P_\tau \tau_t \tag{3.9}$$

and the expectation of the total cycle time is given by

$$E(T_c) = (E(x_1)+1)R_{tt} + E(x_2)R_{tt} + \tau_t P_\tau \tag{3.10}$$

With equations (3.8), and (3.10) we can calculate the throughput T of a TCP flow from equation (3.11) below.

$$T = \frac{E(Q_c)}{E(T_c)} \tag{3.11}$$

This can be achieved with the calculation of the following probability distributions: the loss window probability distribution $P(\sigma)$, the conditional round probability distribution $P(x_1/\sigma)$, the conditional packet count calculation $(Q_1/x_1, x_2, \sigma)$ and the timeout probability calculation τ_t. Further differences in the various packet train models result from the calculation of these distributions. Different TCP versions are also distinguished by these parameters. In the next sections we calculate some of this parameters.

It is good to get a clear picture of the rounds and window evolution in the analysis. The modeled window evolution can be clearly seen from figure 3.1. In figure 3.1 σ_{dn} represents the loss window of cycle n, and x_{ij} represents round j of cycle i. Cycle 1 is in slow start from round x_{11} to round x_{14} from which it enters congestion avoidance to round x_{18}. Here a loss occurs at a loss window of σ_{d1} and results in cycle 2 starting in congestion avoidance with half the window. Cycle 2 grows to round x_{24} where losses occur at loss window of σ_{d2}. However this time the protocol does not recover and results in a timeout with the next cycle starting from slow start i.e.1. The diagram is important in understanding the position of rounds and the packets in the rounds.

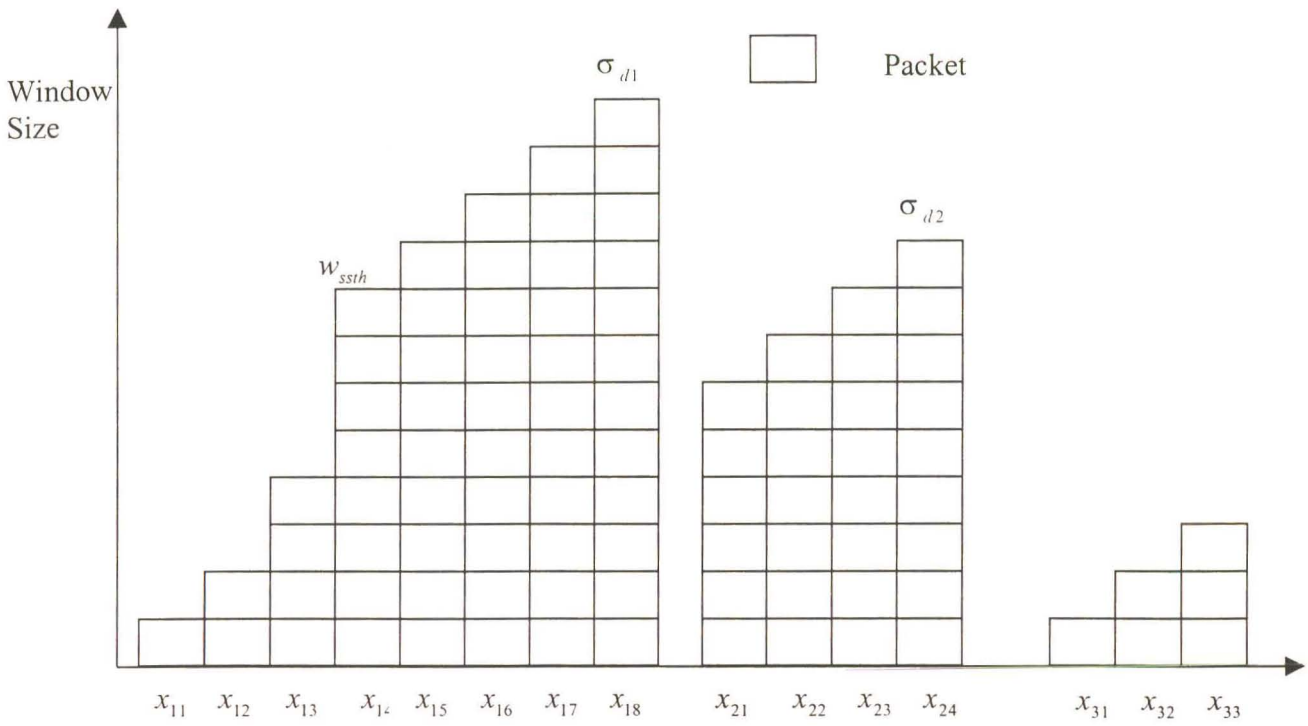


Figure 3.1 TCP window evolution

Note: In this model a packet succeeds with a fixed probability of a and is lost with a probability $b = (1-a)$. Given the round we can determine the window.

3.3.2.1 Loss window probability calculation

In determining $P(\sigma)$ we let U_p denote the maximum window size reached in the p th cycle. The sequences of window sizes at which packets are dropped in successive cycles $\{U_\rho\}_p$ form a Markov chain with transition matrix $P(U_{\rho+1} = \eta / U_\rho = \sigma)$ [30]. The steady state loss window probability $P(\sigma)$, σ = 1, 2, …, W_m can be found from the transition matrix. To get the transition matrix we note that after packet loss the following three scenarios occur: the cycle can begin in slow start and end in slow start, the cycle can begin in slow start and end in congestion avoidance and the cycle can begin in congestion avoidance and end in congestion avoidance. Let $D_{c\sigma}$ be the probability that when loss occurs in a cycle at a congestion window of σ , the next cycle directly starts in congestion avoidance, and then $D_{s\sigma} = 1 - D_{c\sigma}$ is the probability that the next cycle begins

in slow start phase. Let $F_s(\eta)$ denote the probability of a packet loss for a loss window of size η belonging in slow start and $F_c(\eta)$ denote the probability of a packet loss for a loss window η in congestion avoidance. Let $S_s^s(\eta)$ denote the probability of packets successfully reaching the receiver to attain a window size η belonging in slow start starting from slow start, $S_s^c(\eta)$ be the probability of packets successfully reaching receiver to get a window size of η belonging in congestion avoidance starting from slow start and $S_c^c(\eta)$ the probability of packets successfully reaching receiver to get window size of η belonging in congestion avoidance starting from congestion avoidance. The transition probability matrix can be characterized by

$$P(U_{p+1} = \eta / U_p = \sigma) = \begin{cases} D_{s\sigma} S_s(\eta) F_s(\eta) & \text{for } 1 \le \eta \le \eta_{ss} - 1 \\ D_{s\sigma} S_c^s(\eta) F_c(\eta) & \text{for } \eta \ge \eta_{ss} \\ D_{c\sigma} S_c^c(\eta) F_c(\eta) & \text{for } \eta \ge \eta_{ss} \end{cases} \quad (3.12)$$

where η_{ss} is the slow start threshold. We need to calculate the terms of equation 3.12. The values of $D_{s\sigma}$ and $D_{c\sigma}$ will be dealt with in the calculation of the timeout probability. These values are some of the parameters which distinguish between various TCP protocols. Sometimes we drop the η_k notation for the window and use η for clarity.

3.3.2.2 Calculation of $F_s(\eta)$ and $S_s^s(\eta)$

We let η_k be the drop window size during cycle of interest in round k where $0 < \eta_k \le \eta_{ss}$ for slow start and η_{ss} is the slow start threshold. To obtain a loss window size of η_k, $\eta_k - 1$ packets have to be successful, while the η_k th packet has been dropped with a probability b, where k is the sum of the upper limit of $\log_2 \eta_k$ and one, i.e. $k = \lceil \log_2 \eta_k \rceil + 1$. Therefore,

$$S_s^s(\eta_k) = a^{\eta_k - 1}$$
$$F_s(\eta_k) = b \quad (3.13)$$

3.3.2.3 *Calculation of* $S_c^c(\eta)$ *and* $F_c(\eta)$

For us to model window evolution for cycles which don't start from one we let the slow start threshold window size η_{ss} be the starting window and correspond to round one. This is a function of the previous window size. We observe that the window evolution is the same as starting from size one only that it has been shifted up by this value. Let the loss round be k, therefore the loss window is η_k. The value of k is given by $k=(\eta_k+1)-\eta_{ss}$. The successful slow start rounds are given by $k-1$. The loss window size is $\eta_k=(k-1)+\eta_{ss}$. With k and η_{ss} the analysis is similar to the previous section and the probability of packet success for a window in congestion avoidance is given by

$$S_c^c(\eta_k)=\prod_{i=1}^{k-1}a^{\eta_i} \tag{3.14}$$

For drop window of η_k at least one packet has to be unsuccessful. The probability that not a single packet of train with size η_k is dropped is the probability that all packets in the train are delivered. Therefore the probability that at least one packet is lost is given by

$$F_c(\eta_k)=\begin{cases}1-a^{\eta_k-1} & \text{if } \eta_k < W_m \\ 1 & \text{if } \eta_k = W_m \text{ since a packet is surely lost}\end{cases} \tag{3.15}$$

We make an assumption that when the window reaches the maximum size W_m a packet is surely dropped.

3.3.2.4 *Calculation of* $S_s^c(\eta)$

The probability of packets success from slow start to congestion avoidance $S_s^c(\eta)$ is calculated from the above formulas since there exists both slow start and congestion avoidance. We need to know when congestion avoidance starts. Let congestion avoidance start with a window of η_w. The number of slow start rounds is $s=\log_2\eta_w+1$, while the number of congestion avoidance rounds is $l=\eta_k-\eta_w$. We combine the above formulas noting that s is the successful slow start rounds whose probability is given by equation

(3.13), and $l-1$ is the successful congestion avoidance rounds whose probability of success is given by equation (3.14). The total probability is the product and is given by

$$S_s^c(\eta) = S_s^s(\eta) * S_c^c(\eta) \tag{3.16}$$

It should be noted that in this case the packet loss is in the congestion avoidance round.

3.3.2.5 Timeout probability calculation

The packet loss algorithm of figure 3.2 shows the different scenarios of how a time out can occur. A timeout event can be grouped into two groups as below.

- A timeout can be a direct timeout which occurs if less than a certain amount of duplicate ack threshold, Ω (normally 3), arrive at the sender. This is referred to in this thesis as type one timeout.
- It can be an indirect timeout if the TCP algorithm goes to fast retransmit and then the first recovery fails and a timeout occurs. This is referred to as type two timeout in the thesis.

The probability of a timeout P_t is given by

$$P_t = \sum_{\sigma=1}^{W_m} P(\sigma)\tau_\sigma \tag{3.17}$$

where τ_σ is the conditional probability of a timeout in a cycle given that the loss window is σ . $P(\sigma)$ is the loss window probability as calculated in the previous section and W_m is the maximum window size. The value of τ_σ is different for different TCP versions. For old Tahoe a timeout occurs if a packet is lost, type two timeout doesn't exist since there is no first retransmit. For Tahoe the first retransmit behaves as a timeout. Therefore for both Tahoe and Old Tahoe protocols $\tau_\sigma = 1$ in equation (3.17). For Reno and New Reno there is first recovery, both types of timeouts can exist.

3.3.2.5.1 Calculating the probability of a packet loss

A packet drop event occurs with a probability a . Let i represent lost packets, P_i be the probability that i packets are lost and is given by

$$P_i = \binom{\eta_k}{i} a^i (1-a)^{\eta_k - i} \tag{3.18}$$

where η_k is the loss window in round k.

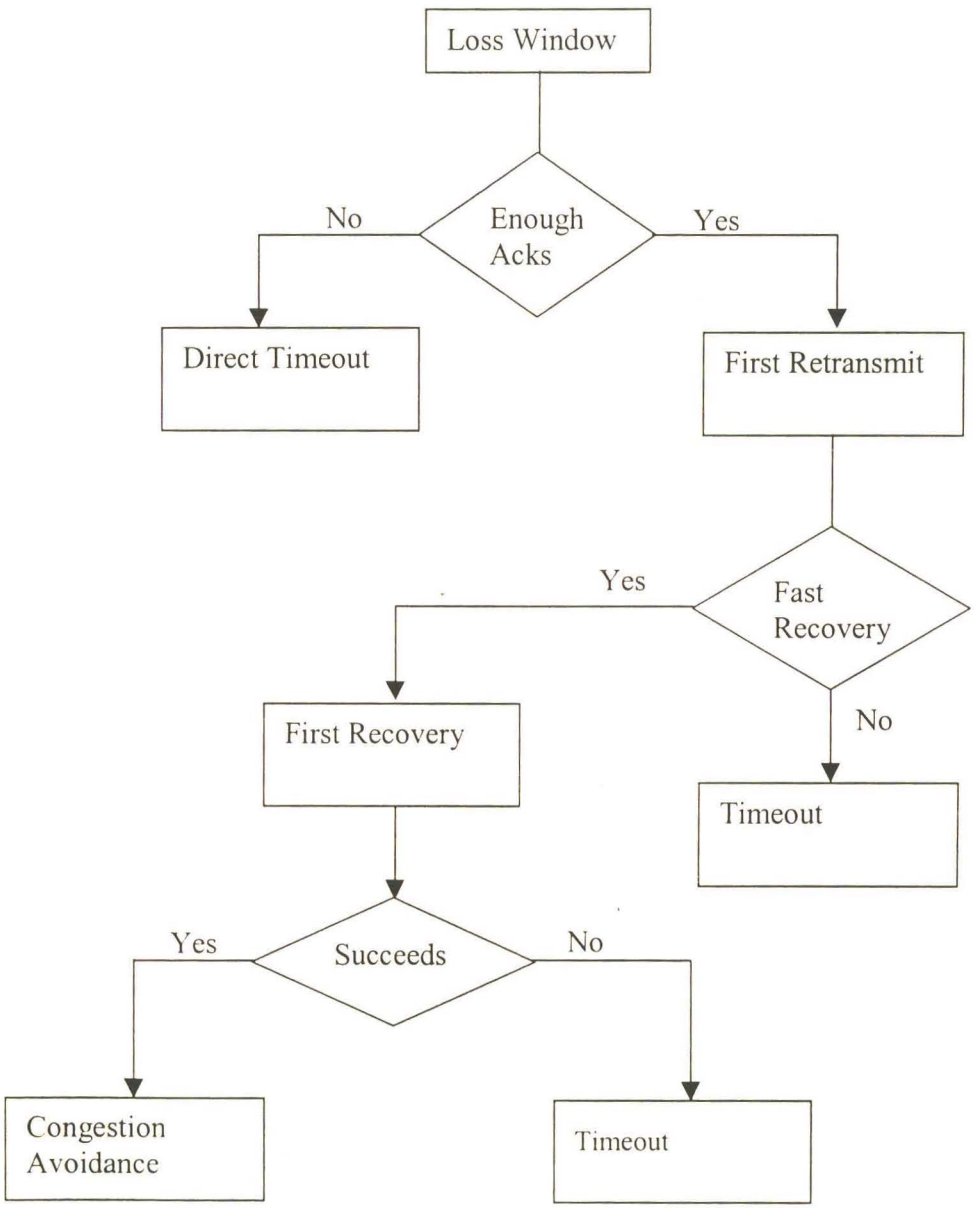


Figure 3.2 TCP packet loss

3.3.2.5.2 Calculating the timeout probability

A direct timeout occurs with a probability P_d if the number of lost packets i is greater than the difference between the loss window and the ack threshold Ω, $i \geq \eta - \Omega$ and is given by

$$P_d = P(i \geq \sigma - \Omega) \qquad (3.19)$$

$$= \sum_{i=\sigma-\Omega}^{\sigma} P_i$$

The indirect timeout occurs only for Reno and New Reno when the cycle goes into first recovery and the first recovery fails. The probability of going into first recovery P_{fr} given by

$$P_{fr} = 1 - \sum_{i=\sigma-\Omega}^{\sigma} P_i \qquad (3.20)$$

For Reno first recovery fails if the following event happen.

- First recovery fails if the first retransmitted packet fails with a fixed probability a.
- It can also fail if the first retransmitted packet succeeds and not enough packets succeed for the second packet to be retransmitted. Let N_d be the number of successful packets after a loss in the loss window η_k. Let n_0 be the number of well-received packets between the two losses. The second packet will be retransmitted if the following inequality in equation (3.21) holds.

$$N_d \geq \left(\frac{\eta_k}{2} - n_0 \right) \qquad (3.21)$$

- First recovery can further fail if the first retransmitted packet succeeds and enough packets succeed for the second packet to be retransmitted and the second packet fails.

The last two conditions are normally simplified to the fact that for Reno first recovery succeeds if the packets lost are less than three [30].

$$\tau_{R\sigma} = \sum_{i=\sigma-\Omega}^{\sigma} P_i + (1 - \sum_{i=\sigma-\Omega}^{\sigma} P_i)\left(\sum_{i=3}^{\sigma} P_i \right) \qquad (3.22)$$

For New Reno first recovery fails under the following conditions.

- First recovery fails if the first retransmitted packet fails with probability as in Reno.
- It can also fail if the first retransmitted packet is successful but second is lost.
- First recovery can further fail if the first retransmitted packet is successful and second is successful but third is lost. This goes on until the number of packets in the window is taken care of.

Thus, a timeout occurs if any retransmitted packet fails resulting in no first ack. Therefore given the number of lost packets in a window we can get the probability that at least one of them fails.

3.3.2.6 Probability of Cycle start $D_{s\sigma}$ and $D_{c\sigma}$

We are looking for the probability $D_{s\sigma}$ that when loss occurs in a cycle at a congestion window of σ, the next circle begins in slow start phase. We note that the next cycle starts in slow start if the previous cycle ended in a timeout. Therefore, $D_{s\sigma} = P_t$ at the loss window σ. The probability $D_{c\sigma}$ that when loss occurs in a cycle at a congestion window of σ, the next cycle directly starts in congestion avoidance is found from $D_{s\sigma} = 1 - D_{c\sigma}$, since the next cycle either begins in slow start or congestion avoidance. It is also given by $D_{c\sigma} = 1 - P_t$. For the various TCP versions we find that

- $D_{s\sigma} = 1$ and therefore $D_{c\sigma} = 0$ for Tahoe and Old Tahoe.
- $D_{s\sigma}$ and $D_{c\sigma}$ are computed from the timeout formulas for the other TCP Protocols.

3.3.2.7 Round probability calculation

We need to find the conditional probability of the number of rounds given the loss window $P(x_1 = m/\sigma)$. For x_1 to be m we require that $m-1$ rounds experience no packet loss and there is at least one packet lost in the m th round. Therefore, the conditional probability is given by

$$P(x_1 = m/\sigma) = S(\eta_m)F(\eta_m) \tag{3.23}$$

where we obtain $S(\eta_m)$ and $F(\eta_m)$ by making the following substitutions in equations (3.14) and (3.15). We substitute m for k in equation (3.14) to get

$$S_c^c(\eta_m) = \prod_{i=1}^{m-1} a^{\eta_i} \tag{3.24}$$

and substitute m for k in equation (3.15) to get

$$F(\eta_m) = \begin{cases} 1 - a^{\eta_m - 1} & \text{if } \eta_m < W_m \\ 1 & \text{if } \eta_m = W_m \text{ since a packet is surely lost} \end{cases} \tag{3.25}$$

From the probability distribution given by (3.23) we can calculate $E(x_1)$ by

$$\begin{aligned} E(x_1) &= E[E(x_1/\sigma)] \\ &= \sum_{\sigma} P(\sigma) \sum_{j} jP(x_1 = j/\sigma) \end{aligned} \tag{3.26}$$

It is possible to model the number of rounds in the loss recovery phase x_2 of the Reno and New Reno algorithms. In calculating x_2 rounds we note that TCP Tahoe doesn't have them. For Reno, we consider that it goes out of fast recovery after a successful retransmitted packet, hence $x_2 = 1$. For New Reno x_2 = number of lost packets, while for Sack $x_2 = 1$. The number of rounds in the timeout interval, x_3, is given by the number of successive timeouts since it takes one round to send a packet.

3.3.2.8 Packet count calculation

Let θ_k be the number of packets in round k, j be the slow start round, η_{ss} the slow start threshold and σ the loss window. We want to find the number of packets delivered Q_1 in the first phase, i.e. until the loss occurs, given the number of rounds m and the loss window σ. It is given by

$$E(Q_1/x_1 = m, \sigma) = \sum_{i=1}^{m} \theta_i \tag{3.27}$$

where θ_k is given by

$$\theta_k = \begin{cases} 2^{k-1} & \text{if } k \le j \\ \theta_{k-1} + 1 & \text{if } j < k \le m \end{cases} \tag{3.28}$$

and $j = (\log_2 \eta_{ss}) + 1$ is the slow start threshold round with the slow start threshold being given by half the previous loss window, $\eta_{ss} = \sigma/2$. This is just counting the number of packets and can be clearly seen in figure 3.1.

3.4 COMPARISON OF ANALYTICAL AND SIMULATION MODELS

The equations (3.1) and (3.2) are the most simplified expressions for TCP. The equations have numerous limitations which are discussed below.

- The equations apply for only one TCP protocol i.e. Reno.
- They only model one source on the network and cannot be used for many sources.
- The equations don't incorporate all the parameters, e.g. first recovery.

Our developed model doesn't have these limitations. Using our developed model, simulation model and the models given by equation (3.1) and (3.2) which are hereby referred to as model 1 and model 2 respectively, we analyse the performance of various parameters on TCP Reno. We simulate a single TCP source with a buffer. The simulation parameters are specified as in the table 3.1, unless explicitly stated.

Table 3.1. Simulation/Analytical Parameters

Parameter	Value
Round trip time	0.01s
Timeout value	0.5s
Maximum window	50
Buffer length	30
Link capacity	2000pkts/s

3.4.1 Effect of Packet loss on TCP Throughput

The first parameter that adversely affects the TCP protocol is the effect of packet losses. TCP responds to the packet losses by reducing the sending rate resulting in wasted bandwidth. The results for TCP Reno protocol in the presence of packet losses is shown in figure 3.3. The maximum window does not affect our analytical protocol. This is

because a maximum window of 50, with a round trip time of 0.01 can be affected by a link whose capacity exceeds 5000 packets per second. However, we have used a link capacity of 2000 packets per second. The throughput is normalised to the link capacity. In figure 3.3 we plot the normalized throughput against the packet loss probability. When the loss probability is less than 0.001 model 1 and model 2 overestimate the throughput. However, their throughput is normalised to the link capacity since the throughput can't exceed the link capacity. When the loss probability is below 0.001, model 1 still overestimate the throughput. However, model 2, simulation and our analysis are fairly close. From the results of figure 3.3, we ascertain that the throughput of a TCP connection degrades when the packet loss probability increases. This is because of the following reasons. Firstly, it takes time to retransmit the lost packets and this wastes bandwidth. Secondly when losses occur, the window reduces and it takes a considerable amount of time for the window to reach the maximum size. Thirdly, in case there is a timeout, the timeout time is quite large.

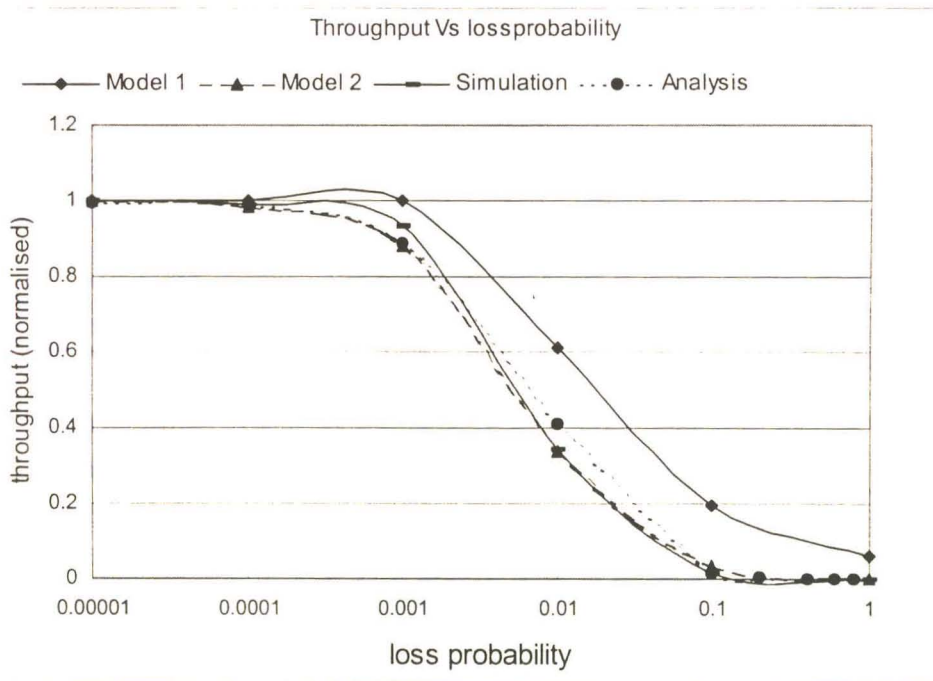


Figure 3.3. Normalized throughput Vs loss probability

3.4.2 Effect of Round trip time on TCP Throughput

Not all sources are located in the same proximity on the network, some can span thousands of kilometres. When these sources deliver their packets to the same destination they are likely to achieve different throughput. In figure 3.4a, we plot the effect of the round trip time on the TCP's throughput at a packet loss probability of 0.0001. In figure 3.4b, we plot the simulation results of the effect of the round trip time on the normalised throughput with varying packet loss probabilities. From figures 3.4a and 3.4b, we conclude that the sources close to the destination are likely to achieve higher throughput than those further from the destination, since throughput degrades as the round trip time increases. This is because of the self-clocking nature of TCP. A close source receives acknowledgements faster and opens its window more rapidly. Model 1 can be looked at as the upper bound of TCP throughput since it overestimates the throughput. Note that model 1 operates well when $p << 1$ [27]. However, the analysis, model 2 and simulation agree and therefore are good models for analyzing the TCP protocol

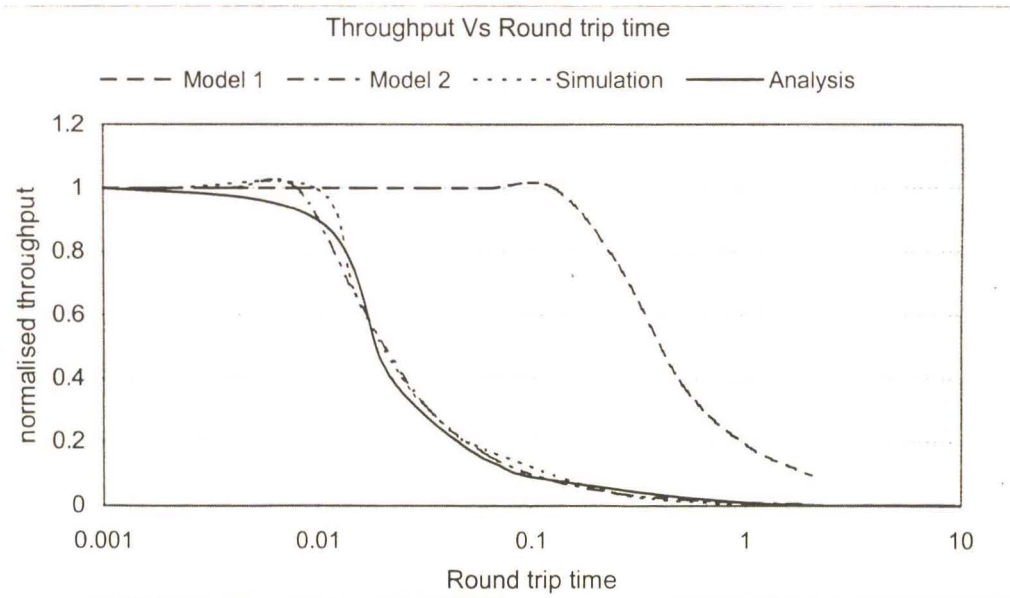


Figure 3.4 a Round trip time

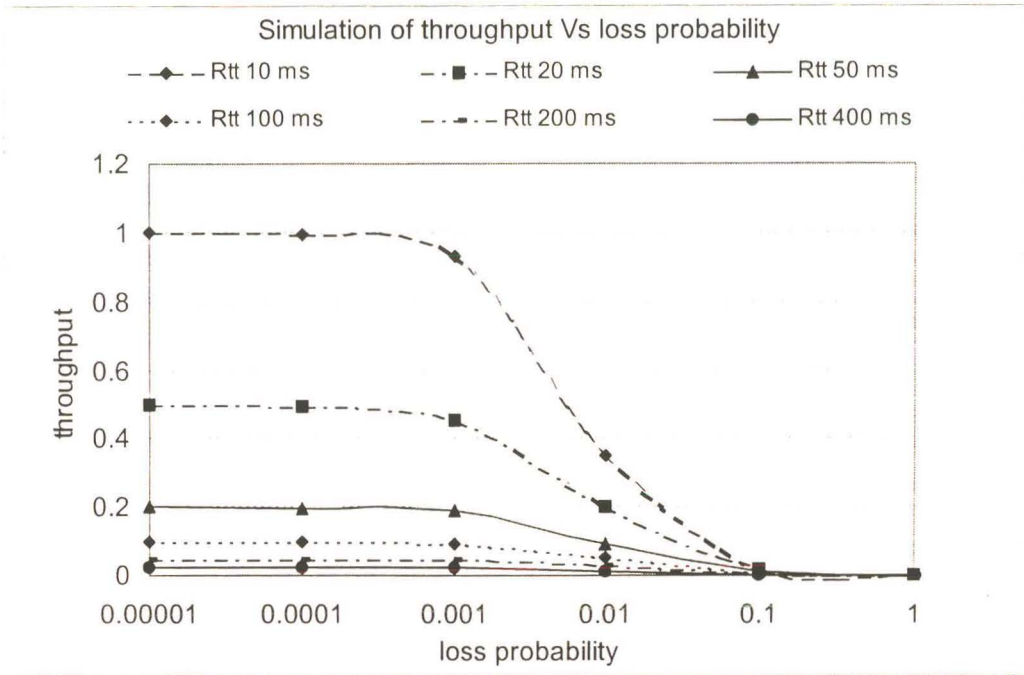


Figure 3.4 b loss probability

Figure 3.4. Throughput Vs Round trip time and loss probability

3.4.3 Effect of Maximum window on TCP Throughput

The maximum window determines the number of packets that can be sent before receiving an acknowledgement. The TCP sending rate is a function of the maximum window and the retransmission time. In figure 3.5a, we plot TCP throughput against the maximum window at a low loss probability of 0.0001. We plot model 2, analysis and simulation. Model 1 does not take into account the effect of the maximum window. From figure 3.5a, we find that the throughput increases linearly with the maximum window until the link capacity is fully utilised at a maximum window of 20 packets. This agrees with the famous conclusion derived in [26], that the throughput is less than the sum of the bandwidth delay product and buffering. The bandwidth delay product is a product of the capacity of the link and the packet transmission time.

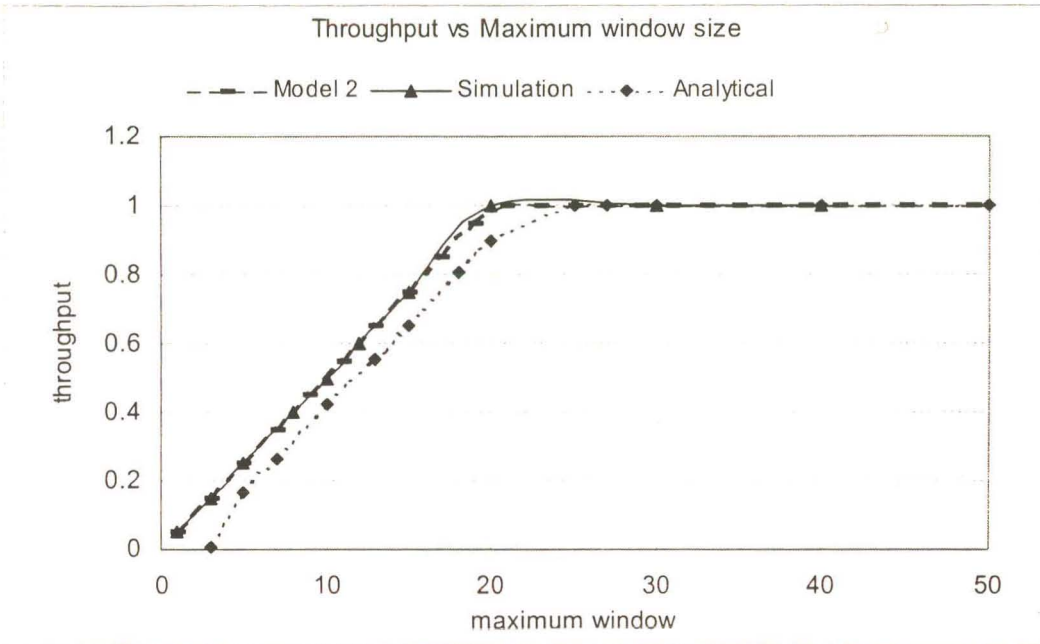


Figure 3.5 a. loss probability

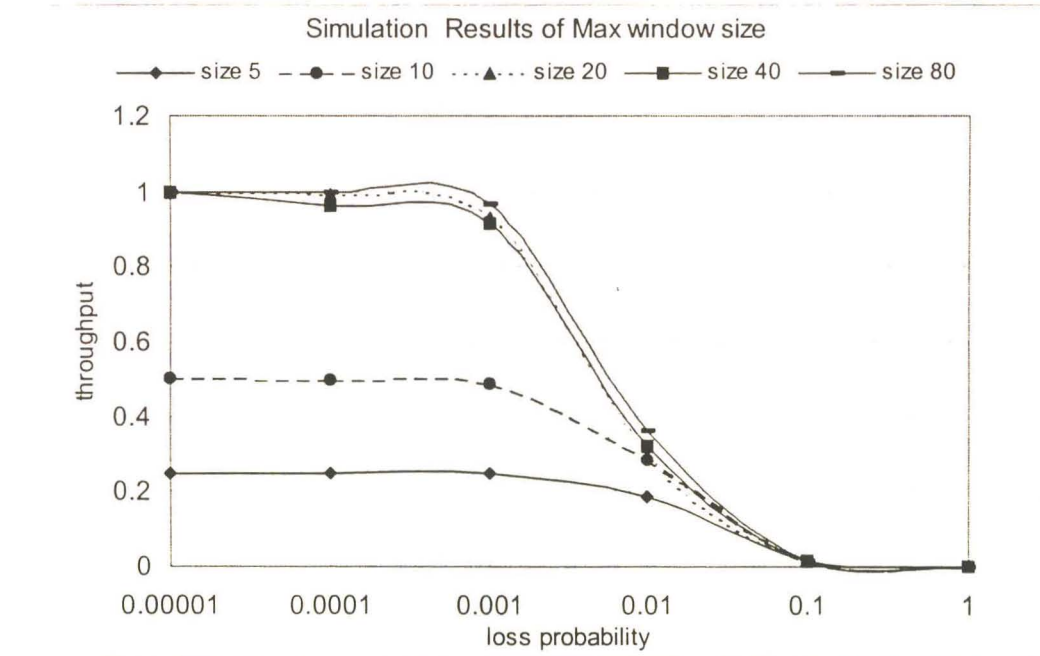


Figure 3.5 b. Maximum window

Figure 3.5. TCP Throughput Vs Maximum window

With a round trip time of 0.01 and a packet rate of 2000 packets per second the maximum window is 20. Any window value above this thus has no effect on the throughput. However, when the window is below this value, our analytical model results are slightly less than the simulation. This is because of the condition imposed that when the window reaches a maximum, a packet is definitely lost. In figure 3.5b, we simulate the behaviour of TCP throughput against the maximum window as we vary the packet loss probability. At a low loss probability of less than 0.001 different sources with different retransmission times receive different throughput, with the source of maximum window of 5 receiving the least throughput, followed by the one with 10. The remaining sources with maximum window of 20, 40 and 80, receive equal throughput. At a high loss probability the effect of increasing the maximum window is nullified by the high losses which reduce the achieved window size and TCP throughput.

3.4.4 Effect of Timeout value on TCP Throughput

TCP fast recovery mechanism fails when there are many losses. TCP uses the timeout mechanism to recover from this condition. Therefore the timeout value is important for TCP performance. We therefore conduct a study to see the effect of the timeout value on TCP throughput. In figure 3.6a, we plot the TCP throughput as a function of the timeout value at different values of the loss probability. The results are plotted for model 2, analysis and simulation. In figure 3.6b, we plot the simulation results of throughput against loss probability while varying the timeout values. From both figures we find that at a low value of loss probability the throughput is unaffected by the increase in the timeout value. This is quite obvious since the protocol hardly times out. There are few losses and since they are not many in a loss window the protocol recovers by means of a first retransmit. As the loss probability increases there are timeouts. And from both graphs the throughput reduces as you increase the timeout value. A large timeout value means the protocol takes more time to start transmitting and hence wasting a considerable amount of bandwidth thus leading to degraded throughput. It should be noted that our model and the simulations and the formula fairly agree in all case.

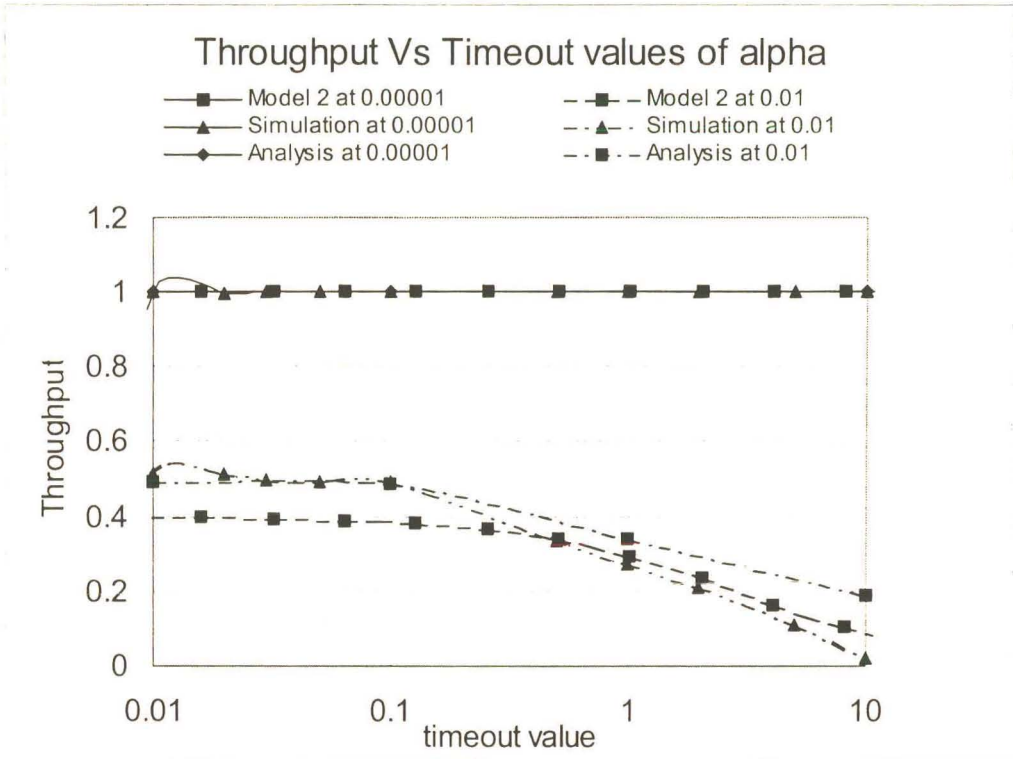


Figure 3.6 a. Timeout

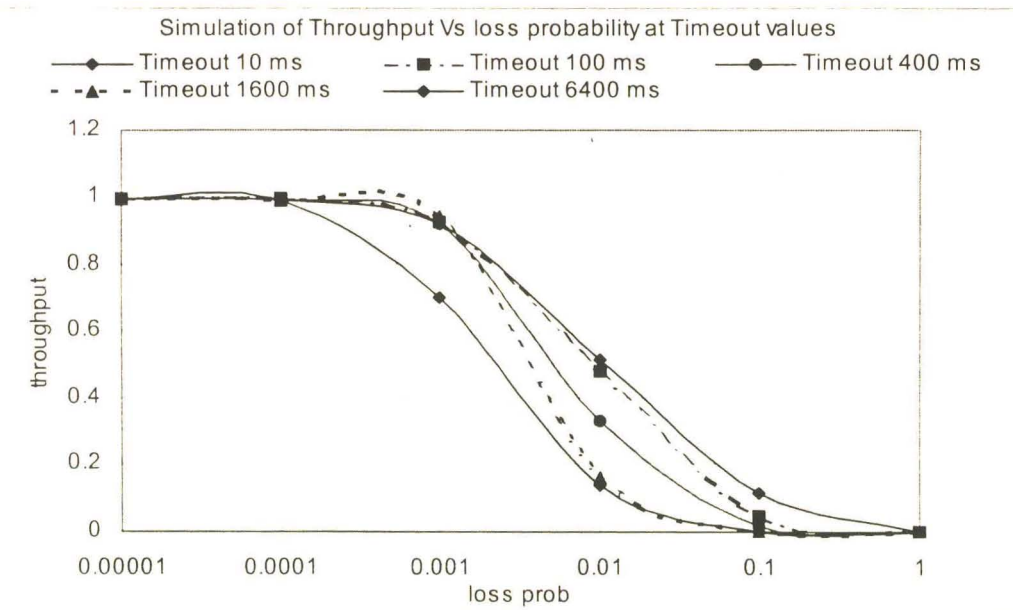


Figure 3.6 b. Loss probability

Figure 3.6. Throughput Vs Timeout value

3.5 SUMMARY

In this chapter we have looked at the performance of the TCP Protocol. We have developed our own analytical model which is robust and can be used to incorporate any parameter of TCP. We compare our analytical model with the two other analytical models developed in literature [27][28] and simulations to analyse the performance of the transport control protocol. We have looked at the effects of the following parameters on TCP performance: packet losses, round trip time, timeout value and the maximum window size. Our analytical results compare favourably with simulation. However, there are slight differences which arise due to the assumption taken. The assumption is that at a maximum window a packet is definitely lost. From the results we have concluded that the performance of TCP degrades with an increase in packet losses. This is because when a packet is lost TCP infers congestion and calls the congestion control algorithm which reduces the window and thus wastes a considerable amount of bandwidth. From the graphs we can infer that TCP performs well when the loss probability is less than 0.001 after which the performance starts to degrade heavily. The performance of the TCP protocol degrades with an increase in the round trip time. This can be attributed to the fact that TCP uses the acknowledgements to increase its window 'self clocking'. The fast acknowledgements open the window faster resulting in higher throughput. The performance of TCP reduces with an increase in the timeout value. When many packets are lost and the TCP window is full, the protocol can only start by means of a timeout. If the timeout value is increased the protocol will wait for a long time before it starts sending the packets and thus wasting a considerable amount of bandwidth. Concerning the maximum window TCP performance improves as you increase the maximum window since the throughput is a function of the window size. However, it reaches a value where it converges since the throughput cannot exceed the link capacity. It should be noted that there are very many other factors that affect the throughput of TCP/IP, these include the IP features of "IP time to live", and IP fragmentation. However we have not focussed on them in the thesis.

CHAPTER 4
PERFORMANCE IMPROVEMENT OF TCP OVER WIRELESS CHANNELS

4.1 INTRODUCTION

TCP was tuned to perform on wire line networks where the channel error rates are very low and congestion is the primary source of packet loss [26]. However, when used over wireless channels which are characterized by high error rates, the performance of TCP is severely affected. Research concerning several aspects of TCP is being conducted widely. Simulation studies of TCP over both wired and wireless networks have been done. Few papers have developed the analytical frameworks. Both the analytical and simulation frameworks fall into two categories, the independent packet loss models and the correlated packet loss models. The work done in [26][30][34] fall in the first category where the packet is lost with a probability p independent of any other packets lost. The other category of papers addresses correlation between packet losses. They include [33][35][36][45]. [28] falls in between since the first packet is lost independently with a certain probability after which all the subsequent packets in the same congestion window are assumed lost. [38] considers packet losses being hidden from TCP by the link level

retransmission mechanism. However, the research done so far has several limitations. Some of them are outlined below. [36] doesn't analyze various versions of TCP, [26] doesn't capture the various TCP phenomena like loss recovery. [37] doesn't capture phenomena like the window limitation and [33] doesn't model fast recovery. In this chapter we develop an analytical model of TCP over a correlated loss wireless channel. In the first part of the chapter we consider the wireless channel models. In the second part we present our analytical model of analyzing TCP over wireless. We then follow by studying the ways of improving TCP Performance over wireless channels with a correlated loss link. Lastly we present some results on improving the TCP protocol by the Snoop and ECN protocol. We also compare our analytical model with simulations for various versions of the TCP protocol.

4.2 WIRELESS CHANNEL MODEL

The average signal strength varies with respect to time and the receiver or transmitter displacement. Depending on the level of variation, radio propagation channels can be divided into large-scale fading and small scale fading. Large-scale fading represents the average signal power attenuation or path loss due to motion over large areas. The statistics of large scale fading provide a way of computing an estimate of a signal's mean path loss and its variation about the mean. Small scale fading refers to the dramatic changes in signal amplitude and phase that can be experienced as a result of small changes (as small as a half-wavelength) in spatial separation between a receiver and transmitter. It is our main focus in this thesis.

4.2.1 Small Scale Fading

The multipath phenomena in the wireless channel results in small scale fading. When the received signal is made up of multiple reflective rays plus a significant line of sight (dominant) component, the envelope amplitude due to small scale fading has a Rician pdf given by equation (4.1), and is referred to as Rician fading [39].

$$p(r) = \frac{r}{\sigma^2} \exp\left[-\frac{r^2 + r_s^2}{2\sigma^2} \right] I_0\left(\frac{r r_s}{\sigma^2} \right) \qquad (4.1)$$

where r is the envelope amplitude of the received signal, r_s is the dominant component $2\sigma^2$ is the predetection mean power of the multipath signal and $I_0(.)$ is the modified Bessel function of the first kind and zero order. As the amplitude of the specular component approaches zero, $r_s \to 0$ the Rician pdf approaches a Rayleigh pdf, expressed as

$$p(r) = \begin{cases} \frac{r}{\sigma^2} \exp\left[-\frac{r^2}{2\sigma^2}\right] & \text{for } r \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{4.2}$$

The Rayleigh pdf represents the worst case of mean fading per mean received power since it has no specular component of the signal.

4.2.2 Quantized Rayleigh Fading Model

It is well known that the effect of narrow-band fading is multiplicative in base-band signal representation.

$$s(t) = u(t) f(t) \tag{4.3}$$

where $u(t), f(t), s(t)$ are the complex envelopes of channel input, fading and channel output respectively. Fading is normally described in the continuous way by a Rayleigh distribution as seen above. The received signal process of equation (4.3) is sampled with a fixed period T representing a time interval which can represent a bit, a symbol or a frame/packet duration. The instantaneous fading power $a(t) = f^2(t)/2$ is quantised with respect to a set of $L-1$ thresholds $\{A_k\}$, $k = 1, ..., L-1$. The resulting process $\alpha = \{\alpha_i\}$ is a discrete process defined as follows

$$\alpha_i = k, \qquad \text{if } a_i = a(iT) \in [A_k, A_{k+1}) \tag{4.4}$$

with $k = 0, ..., L-1$. It has been recently shown [40] that a first order markov model adequately describes the discrete process α_i. The transitional probabilities [41] depend on the average crossing rate of the instantaneous fading power through level A_k. The relevant quantity might not be the channel value $\alpha(t)$, but some function of it, for

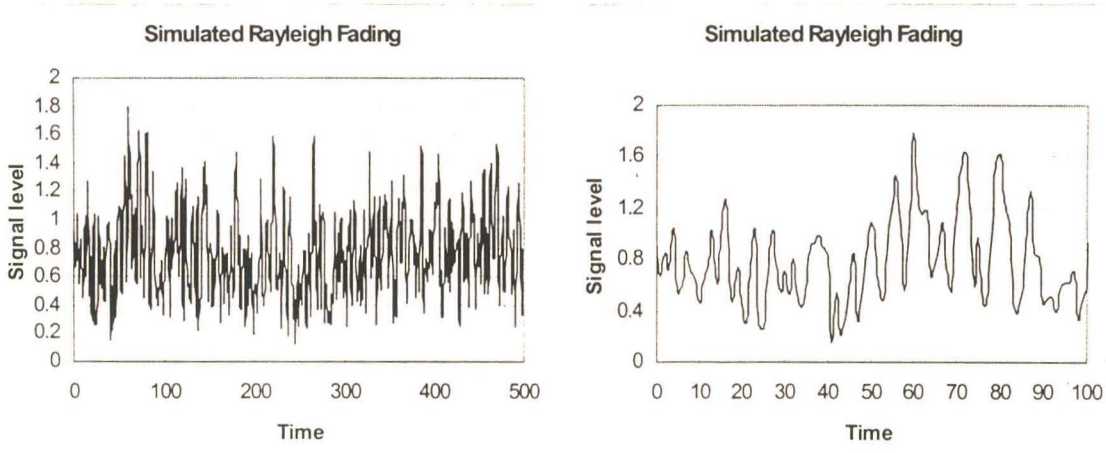
example the error probability of a block of bits, which is a nonlinear function of $\alpha(t)$. In this case, we define a new random process, $\beta(t)$ which depends on $\alpha(t)$, as

$$\beta(t) = \varphi(\alpha(t)) \tag{4.5}$$

and assume an instantaneous memoryless relationship between the two processes. Consider a simplest quantized model with two levels. The two levels can be taken to represent the packet success and failure [42]. The packet condition is determined by comparing the signal power to a threshold (in figure 4.1b it can be taken to be 0.8). If the received power is above the threshold the packet succeeds, otherwise the packet fails. In this case, φ is a step function. The binary process which describes packet successes and failures on the channel β_j, is obtained by quantizing the magnitude of the complex Gaussian description, i.e.

$$\beta_j = \begin{cases} 0 & \text{if } |\alpha_j|^2 > 1/F \\ 1 & \text{if } |\alpha_j|^2 \le 1/F \end{cases} \tag{4.6}$$

where $1/F$ is the power threshold and F is sometimes called the fading margin. 1 stands for packet failure.



a) Larger time scale b) Smaller time scale

Figure 4.1. Simulated Rayleigh fading channel

The first order Markov process works well for modeling a data process when the fading is assumed to be slow ($f_D T << 1$) and the process is thus very correlated. If the values of

$f_D T$ are to be made larger, two samples of the channel are almost independent. For high data rates the fading process is slow and we can't neglect the dependence between consecutive blocks of data. Therefore we cannot use an independent and identically distributed (iid) model to model the relationship between blocks of data. However, the fading is assumed to be slow such that it can be considered constant within a block of data. A number of papers have shown that first order Markovian processes are not adequate for modeling the wireless channel. In [40] they propose the use of more states instead of two states of the markov process. In [43] they present results indicating that the first order Markov chains are not suitable for very slow fading channels. However, they are suitable for very slowly fading applications, which require analysis over a short duration of time. The Gilbert Elliott (GE) is one of the popular first order Markov process for modeling the wireless channel.

4.2.2.1 The GE Wireless Channel Model

The Gilbert Elliott channel is one of the simplest channel models. Here the channel is assumed to be in a good state where the probability of error is small or in a bad state where the probability of error is significantly larger. The GE can be matched to a Rayleigh fading channel by choosing a level for the SNR where the channel is supposed to change state, and then match the average duration the fading amplitude is below this level to the average number of time units the GE channel is in bad state [44]. It can also be matched by comparing the level crossing rate. In the thesis the wireless channel is modeled by the Gilbert Elliott Channel. We follow the modeling as in [45]. It is modeled by a simple two state Markov chain with transition matrix M_c given by

$$M_c = \begin{pmatrix} (1-\alpha) & \alpha \\ \beta & (1-\beta) \end{pmatrix} \tag{4.7}$$

where β is the transition from bad to good, i.e. the conditional probability that a successful transmission occurs in a slot given that a failure occurred in the previous slot and α is the transition from good to bad, i.e. the conditional probability that a successful transmission fails given a success in the previous slot. The average probabilities of packet loss P_E (steady state probability of the channel being in bad state π_B) can be found from

the probabilities in (4.7) by equation (4.10). The probabilities depend on the physical characteristic of the channel which is expressed in terms of the *fading margin, F* and the *normalized Doppler bandwidth,* $f_d T$, where T is the packet duration and f_d is the normalized Doppler frequency whose value is given by

$$f_d = \frac{V}{\lambda}\cos\theta \tag{4.8}$$

$$f_d = f_{dMax} = \frac{V}{\lambda} \quad (\text{maximum doppler frequency})$$

where V is the velocity, λ the wavelength and θ is the angle of approach. Unless otherwise stated we use the maximum Doppler frequency. Different choice of values of P_E and $f_d T$ leads to different models with different degrees of correlation in the fading process. When $f_d T$ is small the fading process is very correlated (long bursts of packet errors). When $f_d T$ is large successive samples of the channel are almost independent (short bursts of packet errors). The average length of burst of errors which is the average duration of bad state is given by $(1/\beta)$ and is described by a geometric random variable while the average duration good state is given by $(1/\alpha)$. For the case where the error process is memoryless with independent and identically distributed (iid) errors equation (4.9) holds.

$$\alpha = \beta = P_E \tag{4.9}$$

However in a correlated loss channel the above parameters are not equal and the steady state probability of channel in good state, π_G, and the steady state probability chain in bad state, π_B, are respectively given by

$$\pi_G = \frac{\beta}{\alpha + \beta} \tag{4.10}$$

$$\pi_B = \frac{\alpha}{\alpha + \beta}$$

For Rayleigh fading channel with a fading margin F the average packet error rate can be found from

$$P_E = 1 - e^{-1/F} \tag{4.11}$$

and the parameter P_{bb}, the probability from bad to bad $(1-\beta)$ can be found from

$$P_{bb} = 1 - \frac{Q(\theta, \rho\theta) - Q(\rho\theta, \theta)}{e^{1/F} - 1}, \tag{4.12}$$

where $\theta = \sqrt{\frac{2/F}{1-\rho^2}}$ and $\rho = J_o(2\pi f_d T)$ is the Gaussian correlation coefficient of two successive samples of the complex amplitude of a fading channel with Doppler frequency f_d, taken T seconds apart [45]. The normalized Doppler bandwidth $f_d T$ describes the correlation in the fading process. J_0 is the Bessel function of the first kind and zero order. $Q(.,.)$ is the Marcum Q function and is given by

$$Q(x, y) = \int_y^\infty e^{-\frac{x^2+w^2}{2}} I_0(xw) w dw \tag{4.13}$$

where I_0 is the modified Bessel function of the first kind and zeroth order.

Note: With P_E and P_{bb} the rest of the parameters can be calculated. In our model we assume that a packet succeeds with a probability of one while in good state and is lost with a probability of one while in the bad state.

4.3 TCP WINDOW EVOLUTION MODEL FOR WIRELESS CHANNEL

We extend the train model of Section 3.3.2 for operation on a wireless channel. We get the expectation of the packets transmitted in a cycle Q_c from equation (3.8) and the cycle time from equation (3.10) and finally the throughput from (3.11). However for the wireless channel we make the following assumptions;

- A TCP packet is transmitted in one wireless slot duration.
- The first packet in a round finds the channel in a steady state condition. Therefore the loss probability of the first packet is different from that of other packets in the round. The loss probability of the next packet is conditioned on the previous packet.

4.3.1 Loss window probability calculation

The steady state loss window probability $P(\sigma)$, σ = 1, 2, ..., W_m is calculated the same way as in Section 3.3.2.1. It is found from the transition probability matrix given by (3.12) and is repeated here for clarity.

$$P(U_{p+1} = \eta / U_p = \sigma) = \begin{cases} D_{s\sigma} S_s(\eta) F_s(\eta) & \text{for } 1 \le \eta \le \eta_{ss} - 1 \\ D_{s\sigma} S_c^s(\eta) F_c(\eta) & \text{for } \eta \ge \eta_{ss} \\ D_{c\sigma} S_c^c(\eta) F_c(\eta) & \text{for } \eta \ge \eta_{ss} \end{cases} \quad (4.14)$$

The differences arise in the calculation of the terms of the equation.

4.3.2 Calculation of $F_s(\eta)$

As in Section 3.3.2.2, η_k is the drop window size during cycle of interest in round k where $0 < \eta_k \le \eta_{ss}$ for slow start and η_{ss} is the slow start threshold. To obtain a loss window size of η_k, $\eta_k - 1$ packets have to be successful, while the η_k th packet has to be dropped. The round number corresponding to $(\eta_k - 1)$ th packet is $k-1$. The position in the round n_s is $(\eta_k - 1) - 2^{k-1}$ i.e. it's the $(n_s + 1)$ th packet in the round, the position of dropped packet $n_d = n_s + 1$. We need to get the probability of a packet being dropped. A packet is either the first in the round or not. We assume the first packet in a window finds the channel in steady state [33]. We consider the events discussed in the section below.

- The first event is that a packet is dropped given the previous one was delivered and it's not the first. This occurs when the channel changes from good to bad. The probability b of this event is given by (see Section 4.2)

  $$b = \alpha \qquad n_s > 0 \quad (4.15)$$

- The second event is that the first packet is dropped. This occurs when you find the channel in bad state and remains there (a packet is dropped) or find the channel in a good state and it changes to a bad state (packet is dropped). The probability a of this event is given by

  $$a = \pi_B(1-\beta) + \pi_G\alpha \qquad \text{if } n_s = 0 \quad (4.16)$$

We know the position of the dropped packet in the loss round $n_d = n_s + 1$. The previous packets were delivered and are taken care of in the calculation of $S_s^s(\eta)$ in the next section. Therefore the probability for a packet loss for a window in slow start is given by

$$F_s(\eta) = \begin{cases} b & \text{if } n_d > 1 \\ a & \text{if } n_d = 1 \end{cases} \quad (4.17)$$

4.3.3 *Calculation of* $S_s^s(\eta)$

Let η_k be the loss window and therefore k the loss round. The number of the fully successful rounds in slow start is given by $(k-2)$. Let P_{all} be the probability that all packets in all the successful rounds succeed and S_{lw} the probability of success in the loss round. All packets succeed when all packets in all the successful rounds succeed and all packets in the loss round succeed. Therefore $S_s^s(\eta)$ is given by

$$S_s^s(\eta) = P_{all} * S_{lw} \tag{4.18}$$

Let P_j be the probability that all packets in round j are delivered, P_{all} is given by

$$P_{all} = \prod_{j=1}^{k-2} P_j \tag{4.19}$$

Let X be the probability of the first packet succeeding and Y the probability of the next packet succeeding. All the packets in a round are delivered if the first packet succeeds and all the next packets in the round succeed. The first packet succeeds if it finds the channel in a good state and is not dropped or if it finds the channel in a bad state and succeeds. The next packet succeeds if the channel changes from good state to good state. The probabilities are given by

$$X = \pi_G(1-\alpha) + \pi_B\beta \tag{4.20}$$

$$Y = (1-\alpha)$$

Let η_j be the window size of round j where $0 < j \leq k-2$. The probability P_j is given by

$$P_j = XY^{\eta_j - 1} \tag{4.21}$$

Combining equations (4.19) and (4.21) gives

$$P_{all} = X^{k-2}\left[\prod_{j=1}^{k-2} Y^{\eta_j - 1}\right] \tag{4.22}$$

Now considering the window in which the packet drops, we know the position of the last successful packet n_s and hence we can find the probability of packets succeeding in this round. This is given by

$$S_{lw}(\eta) = XY^{n_s - 1} \tag{4.23}$$

Therefore with equation (4.22) and (4.23) we can finally find $S_s^s(\eta)$ from equation (4.18)

4.3.4 *Calculation of* $S_c^c(\eta)$ *and* $F_c(\eta)$

As in Section 3.3.2.3 the slow start threshold window size η_{ss} is the starting window and corresponds to round one. The loss window in round k is η_k, where, $k=(\eta_k+1)-\eta_{ss}$ and $\eta_k=(k-1)+\eta_{ss}$. The successful rounds given by $(k-1)$, the analysis is similar to the previous section and the probability of packet success for a window in congestion avoidance is given by

$$S_c^c(\eta_k)=X^{k-1}\left[\prod_{j=1}^{k-1}Y^{\eta_j-1}\right] \tag{4.24}$$

For drop window of η_k at least one packet has to be unsuccessful. The probability that not a single packet of train with size η_k is dropped is the probability that all packets in the train are delivered which is P_k as given by equation (4.21). Therefore the probability that at least one packet is lost is given by

$$F_c(\eta_k)=\begin{cases}1-XY^{\eta_k-1} & \text{if } \eta_k<W_m\\ 1 & \text{if } \eta_k=W_m \text{ since a packet is surely lost}\end{cases} \tag{4.25}$$

with the assumption that a packet is dropped at a maximum window size.

4.3.5 Calculation of $S_s^c(\eta)$

As in Section 3.3.2.4, there exists both slow start and congestion avoidance and we calculate the probability from the above formulas. With the knowledge of the number of successful slow start rounds s with its probability given by equation (4.18), and the number of successful congestion avoidance rounds $l-1$ whose probability of success is given by equation (4.24), the total probability is the product and is given by

$$S_s^c(\eta)=S_s^s(\eta)*S_c^c(\eta) \tag{4.26}$$

Note: there is an implicit difference in the relationship of j and η_j for slow start and congestion avoidance as by the basic TCP algorithm.

4.3.6 Timeout probability calculation

The timeout probability calculation is done similar to Section 3.3.2.5. The difference arises in the calculation of the probability of a packet loss. A packet drop event occurs when the packet finds the channel in bad state. A packet is definitely lost when the channel is in a bad state. The steady state probability that the channel is in a good state π_G and the steady state probability that the channel is in a bad state π_B are given by equations (4.10). The probability that a packet is lost a is given by

$$a = \pi_B \tag{4.27}$$

Let i represent lost packets, P_i be the probability that i packets are lost is given by

$$P_i = \binom{\eta}{i} a^i (1-a)^{\eta - i} \tag{4.28}$$

where η is the loss window. With P_i, further calculations are as in Section 3.3.2.5.

4.3.7 Round probability calculation

The round probability is done the same way as Section 3.3.2.7 Chapter 3. We need to find $P(x_1 = m/\sigma)$. The conditional probability is given by

$$P(x_1 = m/\sigma) = S(\eta)F(\eta) \tag{4.29}$$

We obtain $S(\eta)$ and $F(\eta)$ by making the following substitutions in equations (4.22) and (4.25). We substitute $m-1$ for $k-1$ in (4.22) to get

$$S(\eta) = X^{m-1}\left[\prod_{j=1}^{m-1} Y^{\eta_k - 1}\right] \tag{4.30}$$

and substitute m for k in (4.25) to get

$$F(\eta) = \begin{cases} 1 - XY^{\eta_m - 1} & \text{if } \eta_m < W_m \\ 1 & \text{if } \eta_m = W_m \text{ since a packet is surely lost} \end{cases} \tag{4.31}$$

From (4.31) we can calculate $E(x_1)$ as follows

$$\begin{aligned} E(x_1) &= E[E(x_1/\sigma)] \\ &= \sum_{\sigma} P(\sigma) \sum_{j} jP(x_1 = j/\sigma) \end{aligned} \tag{4.32}$$

Note: The Probability of cycle start, $D_{s\sigma}$ and $D_{c\sigma}$ are calculated as in Section 3.3.2.6 and the packet count calculation is done as in Section 3.3.2.8 Chapter 3.

4.4 PERFORMANCE IMPROVEMENT OF TCP OVER WIRELESS

4.4.1 Current Problems with TCP

TCP operates on any medium regardless of transmission rate, delay, corruption, duplication or reordering of the segments. After the original TCP protocol was introduced the underlying networks have changed. This requires constant development of TCP. TCP faces numerous challenges when used on wireless and other channels.

4.4.1.1 *TCP versus lower layer protocol's congestion control*

TCP runs well over networks, which have only a few features for congestion control and which provide no quality of service. However, if the lower level layer provides congestion control function, problems arise because TCP also provides congestion control. The two congestion control schemes may not work well simultaneously. ATM and X25 are an example of the lower level protocol, which include congestion control. They discard cells and not TCP segments during congestion. This leads to many erroneous TCP segments. All of these segments must be retransmitted by the sender. This increases utilization of the network and decreases efficiency of the connection. It would have been better for discarding entire segments instead of cells and allowing more entire segments to be passed through the switch. Another example is the timeout mechanisms of some protocols like the snoop protocol. If not properly chosen, it would interfere with the operation of TCP.

4.4.1.2 *Quality of Service*

Although TCP provides reliable transmission for applications, it does not guarantee the time within which segments are transmitted from the sender to the receiver. IP datagrams may also arrive in disorder into the receiver in TCP. In other words, TCP does not offer

constant transmission delay and therefore it does not match up with multimedia, audio or real-time applications [46].

4.4.1.3 Asymmetry of the network

End-to-end performance of TCP may suffer due to network asymmetry, especially in the context of wide-area wireless network. This is because full bandwidth may not be achieved in unidirectional transfers due to the slow arrival of acknowledgments. Figure 4.2 illustrates an asymmetrical network.

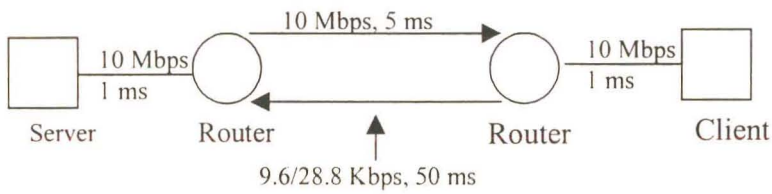


Figure 4.2 Network with asymmetrical bandwidth

Examples of networks that exhibit asymmetry are *wireless cable modem* networks, *Direct Broadcast Satellite* (DBS) networks and *Asymmetric Digital Subscriber Loop* (ADSL) networks. One of the main reasons why interest is increased in asymmetry networks is web traffic, where substantially more data flows from the server to the client than vice versa. As an example, typical TCP connections use a TCP segment size of 1480 bytes giving an IP packet of 1500 bytes. With the policy of acknowledging every other fully sized segment received with the minimal acks it implies that an IP packet of 40bytes is returned for every two 1500 bytes send. We have ignored lower layer headers for illustration. On paths with a 150:1 of forward bandwidth to reverse bandwidth, the reverse path becomes a bottleneck and prevents fully utilization of the forward link. As the segment size reduces the problem sets in at a much lower ratio.

4.4.1.4 *Dependence of window growth on RTT*

The TCP's self-Clocking mechanism leads to unfairness between connections traversing widely differing paths in the network. Connections with shorter RTT can increase their rate of sending more rapidly, and so end up capturing most of the networks bandwidth at

the expense of long delay networks since their clock runs faster. This dependence has been seen in the results of Section 3.4.2.

4.4.1.5 Limited Receiver Window

The receivers offered window is represented in the TCP header by a 16-bit field, which restricts its value to 64 kilobytes. The window size might be limited even further by some implementations. This limitation reduces throughput over paths with high bandwidth delay products, such as geostationary satellite links, as TCP can only transfer one window of data every RTT. The dependence of TCP on the maximum window has been seen in Section 3.4.3.

4.4.1.6 Congestion Loss and High bit Errors

TCP was tuned to perform on wired networks like copper of typical bit error rate (BER) on the order of 10E-6 to 10E-8 and therefore packet loss is due to congestion. Wireless channels are characterized by losses due to transmission channels and handoffs. They have high BER, usually of the order of 10E-3 and can go as high as 10-1. TCP interprets the wireless losses as congestion and invokes the congestion control mechanisms resulting in degradation of performance. When a packet is lost due to bit errors the sender's window is halved even though there is no congestion in the network and thus underutilizing the network.

4.4.1.7 Others

The other problems of TCP are outlined as below.

- The wireless networks are limited in bandwidth. See Section 1.2.
- The users are mobile and these results in handoff problems.
- Short flows like email and web browsing finish their transmission while sender is still in slow start. This wastes bandwidth.
- Some application layer protocols are poor. They don't consider the operation of TCP e.g. those that open many TCP connections.

4.4.2 Methods of improving TCP over Wireless

Wireless networks have the several alarming issues. They have high bit error rates, are limited on bandwidth, and have problems brought about by the mobility issues. TCP cannot make efficient use of satellite channels. This is because of their characteristics as discussed below.

- Satellite channels have a large bandwidth delay product. The bandwidth delay product defines the amount of data at any time on the link for maximum utilization of the available bandwidth.
- Satellite channels have a long feedback loop. Therefore it takes the sender quite some time to know whether the packet was lost or received.
- Satellite channels are disadvantageous for interactive applications like telnet and congestion control algorithms.
- They have a high BER which poses a problem to TCP by throttling the transmission window.
- They have variable round trip delays. In low earth orbit constellations the delays varies from time to time, this affect TCP as seen by the fact that it behaves differently with different round trip times.
- Handoffs from one satellite to another cause losses, this affects TCP.
- Some connections use asymmetrical Links, they use satellite links for one direction and terrestrial link for opposite direction, which poses a problem for TCP.

With all these characteristics and the limitations of TCP as discussed above we definitely conclude that TCP throughput is highly reduced when used on satellite and ordinary wireless channels. Therefore for TCP to be effectively used on the wireless channels especially with the growth of web based Internet applications, its performance needs to be improved. There are several proposed methods for improving TCP performance over a wireless channel. The aim of most of the methods of improvement of TCP over wireless is to separate congestion control from packet losses since TCP throughput is reduced by misinterpreting wireless losses for congestion losses. They can be classified into the following categories [47]. End to end, link layer schemes, split connection schemes and the other methods. These are discussed in the following sections.

4.4.2.1 End To End Mechanisms

End-to-End mechanisms [25] involve changes on the TCP protocol at the host and destination side. This includes the various TCP protocols that enable it to recover from multiple losses in a window. This various algorithms have been discussed in Section 3.2. The others include the TCP forward acknowledgement [72]. This aims at estimating intelligently the amount of data transmitted and unacknowledged and thus makes an intelligent estimation about the data that should be retransmitted. Another is the TCP Santa Cruz [73]. It keeps time when the segment was sent and received and hence determines congestion. It alleviates the problems of network asymmetry, limited bandwidth and variable delays. Another End–to-End method is the Explicit Congestion Notification (ECN) which will be discussed latter.

4.4.2.2 Link Layer Schemes

These try to transparently enhance the quality of wireless links through forward error correction, local acknowledgement, and selective retransmission. Therefore a normal TCP protocol sees a wireless link as close to a wired link as possible. They include the link layer protocols and the Snoop Protocols. In this section we look at the link layer schemes. The Snoop Protocol will be discussed in Section 4.5.4. In order to reduce the FER seen by TCP a Radio Link Protocol (RLP) is used at the link layer in cellular communications [24]. An example is in DS-CDMA where an RLP with a negative acknowledgement has been standardized (IS-95). The RLP performs a partial link layer error recovery through a limited number of RLP frame retransmissions in case of frame error. The RLP is a pure negative acknowledgement (NAK) based selective repeat protocol implying the receiver does not acknowledge correct RLP data frames. Since the number of allowed retransmissions is finite, RLP cannot completely eliminate all detectable errors. These are passed on to TCP. GSM also employs nontransparent mode RLP which uses selective repeat automatic request. These link layer methods have proved effective since they don't disorganize the protocol layering.

4.4.2.3 Split TCP Schemes

The Split TCP schemes are commonly called Indirect TCP [48]. Split connection schemes involve splitting a TCP connection between a fixed and a mobile host into two separate connections, one for the wired section and the other for the wireless section. A more optimized wireless link specific protocol is used on the one hop wireless link for better performance. This achieves a separation of flow and congestion control of the wireless link from that of a fixed network. The drawbacks include the fact that the TCP protocol is no more end-to-end. There is application relinking at the base station and there is an increase in software overhead.

4.4.2.4 Other Mechanisms

4.4.2.4.1 Large Initial window

The slow start algorithm gradually probes the network for available bandwidth. It wastes bandwidth especially during startup and when used on long delay networks. An initial large window reduces the amount of time required by slow start [49]. An experimental TCP extension outlined in [50] allows the initial window size to be increased from 1 to that given by

$$\textit{Minimum of (4*MSS, maximum (2*MSS, 4380 bytes))} \tag{5.6}$$

where *MSS* is the maximum segment size of a TCP segment. An increased window implies that more packets are sent during the first RTT of data transmission triggering more acks hence window opens rapidly to utilize the available bandwidth at startup. This is very effective on satellite channels with a large bandwidth delay.

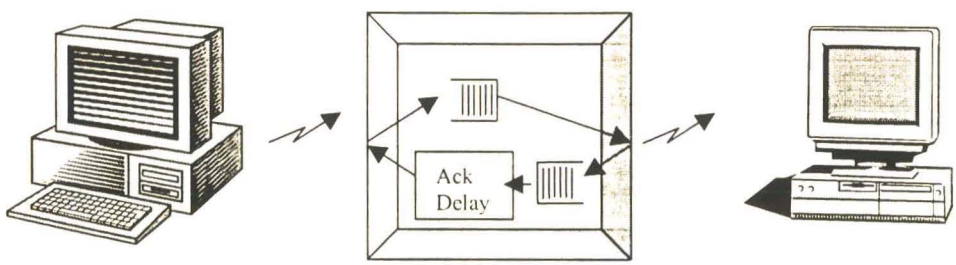


Figure 4.3. Fast TCP

4.4.2.4.2 Fast- TCP

This is a TCP acknowledgement delay scheme [51]. The network element, such as a router, delays the IP packets carrying TCP acknowledgements when network congestion is about to occur. The TCP source which is not receiving acknowledgements keeps its current transmission window until a delayed acknowledgement is received. The router monitors the traffic in the forward direction and controls the acknowledgement rate hence controlling the window growth rate. This scheme is as shown in figure 4.3, it does not modify the transport protocol. It operates at IP level and requires a modification at this level.

4.4.2.4.3 The Fast Retransmit approach

This solves the problem of handover losses in the network [52]. Since the handover delay is looked upon by TCP as a loss, it goes into first retransmit without waiting for a timeout which considerably takes time and wastes the networks resources. Hence recovery is initiated immediately after handoff.

4.4.2.4.4 TCP/IP header compression

It has been noted that many fields in the TCP and IP headers (e.g. source and destination address) remain constant during the course of a session. The proposed TCP and IP compression reduces overhead by replacing these data that remains constant, changes slowly or change in a predictable manner with a short connection [52]. The sender first sends a full TCP/IP header, including in it a connection number (short connection) that the sender will use to reference the connection. The receiver stores full header and uses it as a template, filling in some fields from the short connection.

4.4.2.4.5 Decoupling Approach

Decoupling Approach [53] is a new method that has been proposed. Here the headers are send separately from the data packets. The frame error rate is proportional to the packet

size. The headers packets of a control TCP are send under congestion control. They are of smaller size relative to TCP packets hence experience less non congestion losses, i.e. FER is smaller. For each transmitted header, a proportional amount of data is sent in the data circuit. Hence the TCP is also under congestion control. The assumption being that the header and the data packets follow the same path.

4.5 PERFORMANCE OF VARIOUS TCP PROTOCOLS OVER WIRELESS CHANNELS

In this section we look at the performance improvement of various TCP Protocols over a wireless channel.

4.5.1 Simulation Model

We conducted a simulation to investigate the performance of various TCP protocols on the wireless channel. For our simulation model we consider infinite data TCP sources, which always have packets to send. Hence the units of data are maximum (1400bytes). We consider a wireless link of capacity 1.5Mbps; this implies a packet transmission time of 7.5ms on the wireless link. A FIFO buffer of capacity B packets as in figure 4.4. The wireless channel is modeled as the Gilbert Elliott Channel by a simple two state Markov chain with transition matrix M_c given by equation (4.7).

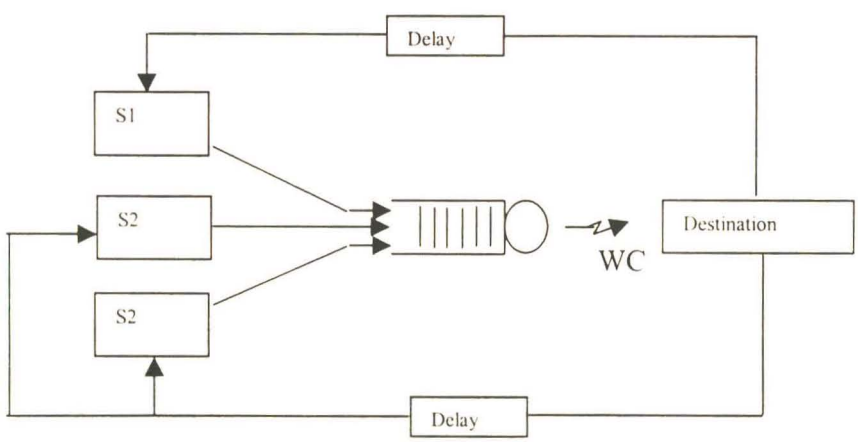


Figure 4.4 Simulation model

4.5.2 Results and Discussion

In our simulation, times are normalized to the mean packet times, the timeout granularity chosen as 50. The normal first retransmit threshold of 3 is used for both TCP versions. The buffer size used is 30, corresponding to buffer of approximately 40kB with the packet sizes as given. The value of β was taken to be 0.1 whereas α was varied in the simulation. In figure 4.5 we plot the normalized channel throughput versus the packet error probability for different TCP versions.

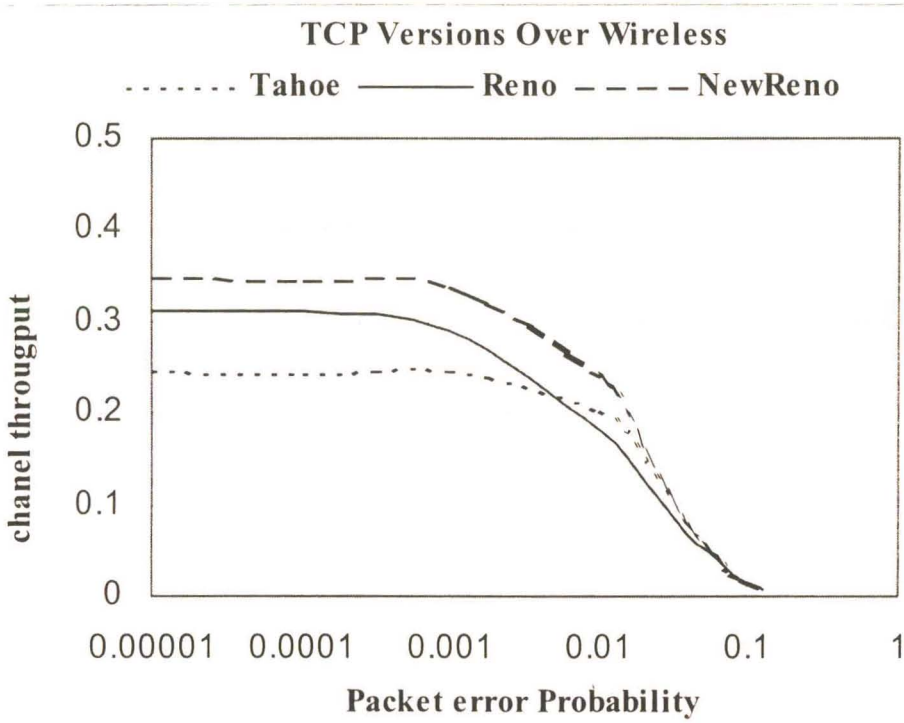


Figure 4. 5. TCP Tahoe, Reno, NewReno over wireless

The results show that channel utilization/throughput degrades with increasing packet error probability with different TCP versions behaving differently. As previously seen TCP Tahoe performs the worst for an error probability less than 0.01 since there is no first recovery. But when the loss probability increases beyond 0.01 first retransmit does not succeed very often especially for Reno and hence Tahoe performs better than Reno. New Reno is superior in its performance to the other two. The TCP versions behave differently as parameters such as the retransmission time, window sizes and others are

varied. TCP protocols are continually being improved in performance with newer protocols like SACK and Vegas coming up which perform better than the older protocols.

As well known the TCP throughput is a function of the TCP window [20]. Therefore for the simulation done above we plotted the TCP windows for the three protocols at timeout and first retransmit instants. The results are shown in figure 4.6. At a low value of alpha, figure 4.6a, the window sizes are high. They are almost of the same size with the New Reno window being the highest. The intensity of the points is less for New Reno showing it experiences the least number of timeouts and first retransmits. This is because New Reno responds to losses better than Reno and Tahoe. It always has packets in the buffer and therefore rarely times out. Thus the overall throughput is high. Note that the peaking of the New Reno window is due to the increment of the window in fast recovery as more packets go through. At the medium value of alpha, figure 4.6b, the New Reno window is still high, relatively followed by Reno and then Tahoe. At the lowest value of alpha, figure 4.6c, the window sizes have been drastically reduced. There is no marked difference between all the protocols. The intensity for timeouts and first retransmit is almost the same for all the protocols and it is high. This is because the losses are too much and all the protocols recover mostly by timeouts.

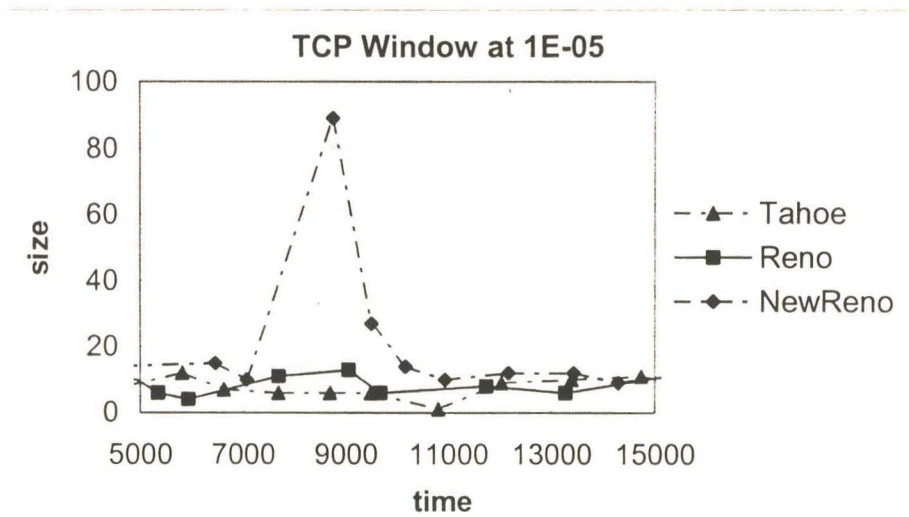


Figure 4.6a. Window at lower alpha

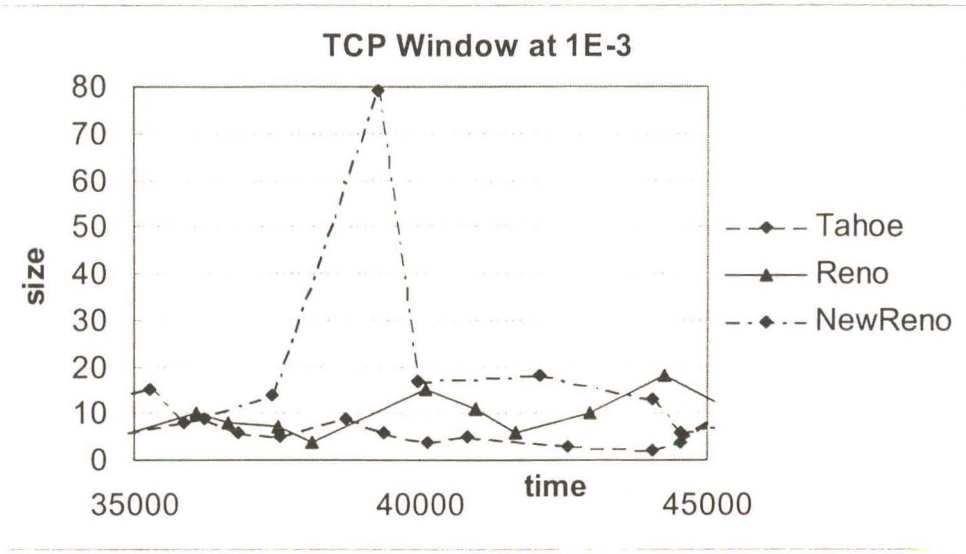


Figure 4.6 b. Window at medium alpha

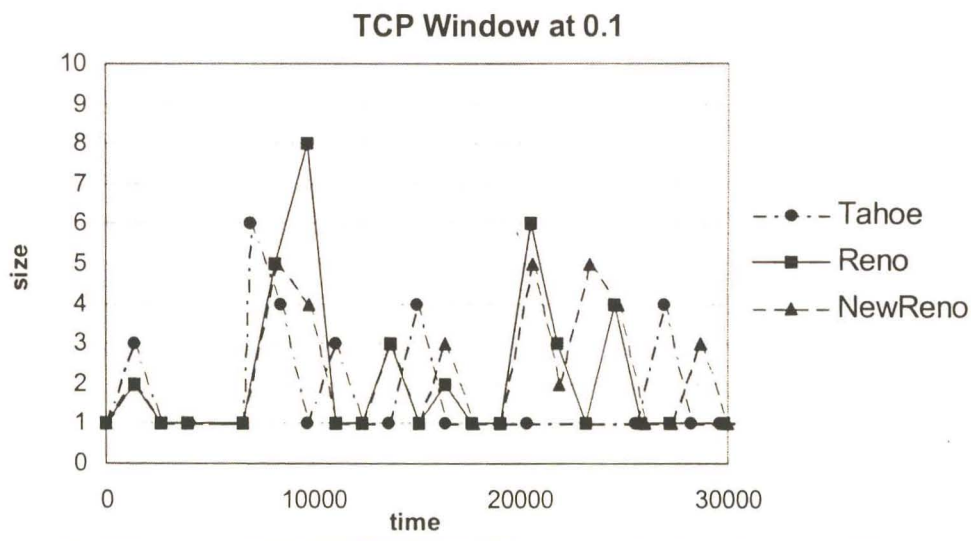


Figure 4.6c. Window at high alpha

Figure 4.6. Window at various values of alpha

4.5.3 Comparison of Analytical and Simulation Model

The developed analytical model was compared to the simulation model and the results plotted as in figure 4.7. We compared TCP Reno and Tahoe on a wireless channel with parameters as in the table 3.1 Chapter 3. However, the maximum window was taken to be 20 packets. The results as earlier seen show that the throughput of both Tahoe and Reno degrade as the wireless losses increase. As the wireless losses increase Tahoe starts performing better than Reno. The simulation results show a slightly lower value of the wireless losses where Reno starts performing worse than Tahoe. Generally our analytical model results are slightly higher than the simulation model. Apart from the minor differences with the simulation we can deduce that we model TCP over wireless accurately with our developed model.

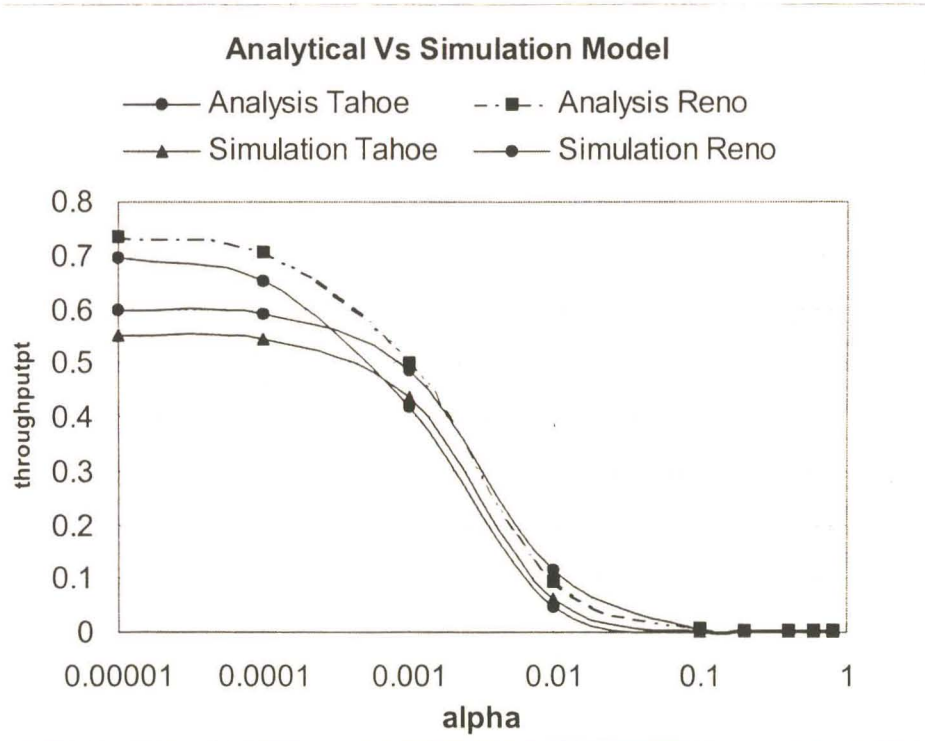


Figure 4.7. Analytical and simulation model comparison

4.5.4 Performance of Snoop and ECN TCP Protocols over Wireless channels

In this section we look at two protocols for improvement of TCP over wireless. These are the Snoop and ECN protocols. The snoop and ECN protocol have been implemented in [25]. However, they have not been emulated on the same network. Furthermore the wireless losses have been artificially generated since they know the loss distribution. They have not taken into consideration the RED like algorithm for marking. The RED algorithm is the favorite marking algorithm for ECN. We introduce the modeling of both protocols on the same network with a correlated loss link where our ECN protocol employs a RED like algorithm for marking.

4.5.4.1 The Snoop Protocol

The snoop protocol [47] involves adding to the base station a module called the snoop agent. A snoop agent tracks TCP data and acknowledgement. It caches unacknowledged TCP packets and uses the loss indications conveyed by the duplicate acks to retransmit the lost data. By doing this the base station hides the losses from the host by not sending the acknowledgements and hence triggering congestion control mechanisms. The snoop protocol operates by two linked procedures, snoop data and snoop ack. The snoop data procedure passes data to the mobile host. The snoop data using sequence numbers is in charge of caching packets if they haven't been cached earlier and forwarding the packets to the destination. The snoop ack on the other hand monitors the ACK's sent by the destination and performs various actions. If the ACK is new, it forwards it to the sender. If it is a duplicate ACK it retransmits the lost segment.

4.5.4.2 Explicit Congestion Notification Protocol

ECN [54] allows routers to inform TCP senders about imminent congestion without dropping segments. Two variants have been proposed, backward ECN, where the router transmits messages directly to the sender as an Internet control message protocol (ICMP) source quench message indicating the TCP sender should reduce its sending rate. The other is the forward ECN. In forward ECN when the receiver receives a marked

congestion packet it informs the sender in the acknowledgement about the incipient congestion and the sender reacts by invoking congestion control. ECN requires the support from both the routers and the sending and receiving TCP hosts. The ECN capable router side modifies the RED algorithm. Instead of dropping the packet, it marks the packet by setting IP Congestion Experienced (CE) bit of the packet header. This is bit 7 of the traffic class (Ipv6) or of the type of service (Ipv4). The router only marks the packets that are ECN capable. This is achieved by the sender setting ECN capable transport bit (ECT) for ECN capable packets. This is bit 6 of the traffic class (Ipv6) or of the type of service (Ipv4). The host TCP process is as follows. At connection setup the host sets the ECN-Echo (ECE) flag and congestion window reduced (CWR) flag in the SYN packet to show desire of using ECN. Bit 9 in the reserved field of the TCP header is designated, as ECE while bit 8 is CWR. The receiver sets the ECE flag in the SYN-ACK in response. The sender then sets the ECT bit in all the packets. If the sender receives an ack indicating congestion it takes appropriate steps and then sets the CWR bit on the next outgoing packet. This informs the receiver that it has reacted to congestion and hence the receiver stops sending congestion notification if there is no new congestion. The receiver TCP's duty after connection setup is to check the CE bit in the received packet and sets the ECN- Echo flag (a flag send by the receiver to indicate congestion, bit 9 of reserved TCP header). The receiver uses the CWR to determine when to stop setting the ECE flag. Note: See Appendix 1 and Appendix 2 for TCP and IP headers.

4.5.4.3 Simulation model

The simulation model is similar to the one in figure 4.4, except that we have two sources. The first source is the normal TCP which employs Snoop protocol and we refer to it as Snoop TCP. The second source is the one employing the ECN protocol. Both the sources have infinite data hence the units of data are maximum (1400 bytes). The wireless link is modelled by the GE channel as before. The buffer employs a RED algorithm. A snoop packet is put both in the buffer and the cache buffer from where it is removed if successfully transmitted. For the ECN protocol we make a slight change. The ECN packets are also dropped by the RED. When a packet is dropped the router sets a Congestion flag. If the congestion flag is set the router marks the CE bit of the packets.

4.5.4.4 Discussion and Results

The parameter settings unless otherwise specified for this section are like the previous section. In figure 4.8a, we plot the behaviour of the snoop TCP protocol versus the original TCP protocol. We choose to plot the results for TCP Reno since the other protocols exhibit almost the same behaviour. The results as predicted indicate that the snoop protocol performs better than the original protocol until the packet error rates become excessively much at 0.1 and TCP relies heavily on timeouts as a means of packet loss notification. At this moment they both behave the same since packet loss is mainly detected by timeouts. Similarly in figure 4.8b, we plot the behaviour of ECN TCP versus the original protocol. The results are similar to the snoop and original protocol. In figure 4.9 we compare the performance of the improved protocols, ECN and the snoop protocol. From the graphs we see that for all the three versions of TCP protocols the snoop protocol performs better than the ECN protocol up to an error of about 0.1 after which they start behaving the same way. This is the time at which packet losses are detected by timeouts hence the protocols behaving the same way. For the snoop protocol graphs the performance of New Reno is better than that of Reno whose performance is better than Tahoe as shown by the comparison of the original protocols. The same scenario happens for the ECN scheme as well.

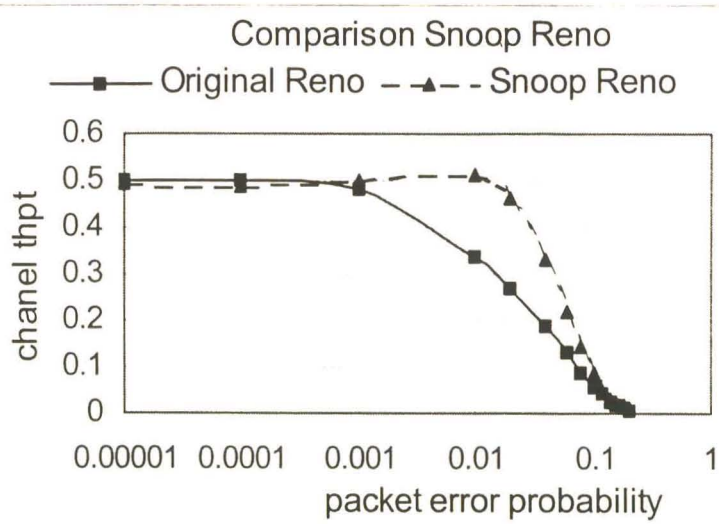


Figure 4.8a. Comparison of snoop and original protocol

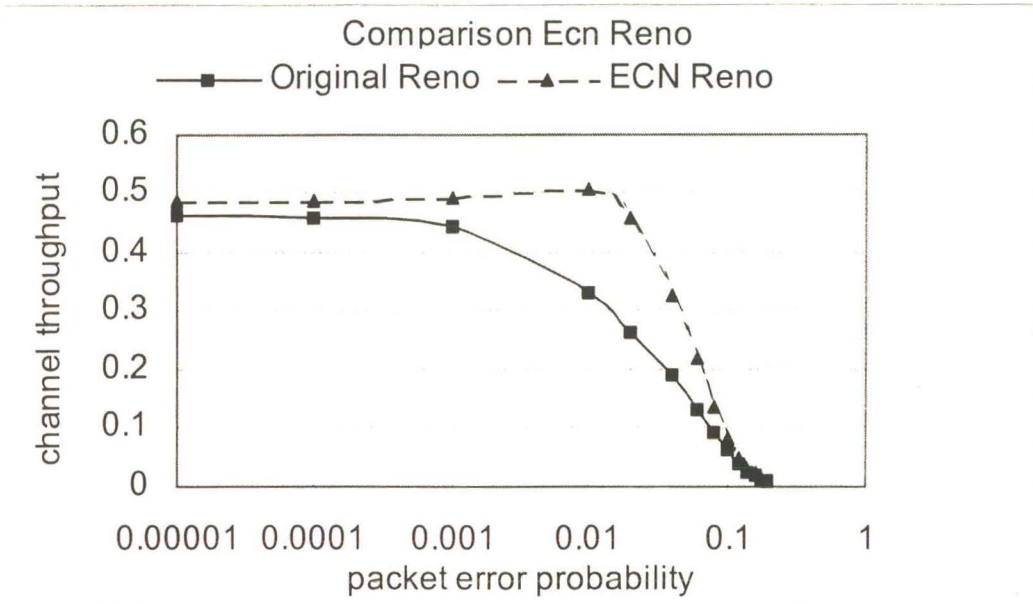


Figure 4.8b. Comparison of ECN and original protocol

Figure 4.8. Comparison of Improved Vs Original Protocols

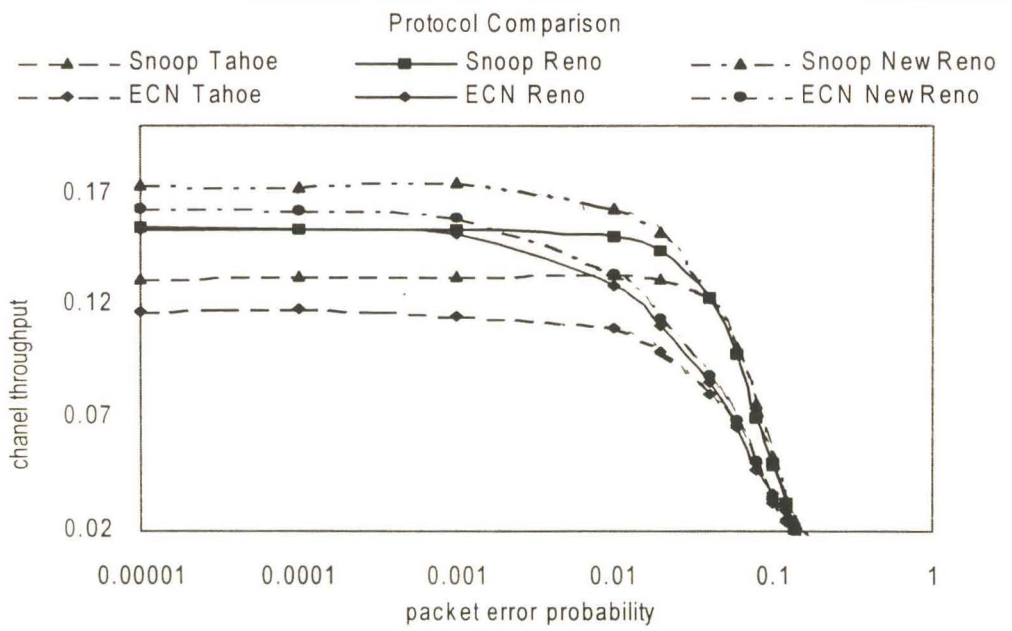


Figure 4.9. Comparison of snoop and ECN all protocols

4.6 SUMMARY

In this chapter we have looked at the TCP Protocol and its performance over a wireless channel. We have looked at the performance of TCP versions over wireless channels and the ways of improving the performance of TCP over wireless channels which we have classified into End-To-End, Link Layer Schemes, Split connection Schemes and the Other methods. In particular we have compared two of the methods of improvement of TCP over wireless. These are the Snoop and the ECN protocols. We have found that both perform better than normal TCP on a wireless channel. However, Snoop protocol performs better than ECN protocol. We have developed our analytical model for analyzing TCP over wireless. We have compared it with simulation for TCP Tahoe and Reno and observed that it performs well as compared with the simulation. It is a more robust model that can be adapted to model any aspect of the TCP protocol. We have comprehensively put together the full behavior of TCP over wireless channels.

CHAPTER 5
TCP OVER WIRELESS WITH DIFFERENTIATED SERVICES

5.1 INTRODUCTION

The previous chapters have laid a foundation on TCP, the differentiated services model and the wireless channel model. In this chapter we combine the performance of TCP throughput on a network with differentiated services. This is done over a wireless channel with correlated packet errors modeled by a Markov chain. We extend the analytical model for TCP over wireless to TCP over wireless with differentiated services. We model the assured differentiated services schemes in the network and look at the way it behaves with several TCP sources. The analytical framework results are validated by simulation results. DiffServ has been studied in literature: [55] quantify the expected delay and loss of packets that arrive via a Poisson process for assured and premium service. In [68] they analyze various router and packet mechanisms to satisfy session requirements. [56] deals with quantifying the throughput while [22] considers relative and absolute differentiated services. TCP has also been studied in literature. In all these cases no paper has analyzed TCP with differentiated services except [34]. However the paper doesn't look at a

wireless channel with correlated errors. The errors are only due to congestion. And a packet is lost with a fixed constant probability. With the deployment of the QoS mechanisms like RED in the Internet routers we need an analytical model that can incorporate them in the analysis. From these there is need for an analytical model of TCP over wireless with differentiated services. These can provide an insight in improving the performance of TCP over wireless with DiffServ. In these chapter we derive an analytical model for TCP over wireless with Differentiated services. Our analytical model if finally proved by a simulation model of TCP over wireless with differentiated services.

5.2 ANALYTICAL MODEL

5.2.1 Model Overview

In our model as in [26] we consider TCP data sources which generate a Poisson distributed number of packets of maximum size (1400 bytes). We consider various classes of TCP connections running on a wireless link assumed to be the bottleneck with capacity μ packets per second with each class containing several sources. We model a network with assured services. Each packet of a particular service class is marked as IN or OUT depending on reservation rate. Both IN and OUT assured service packets are then forwarded into the buffer of which we will henceforth refer to as the congestion buffer.

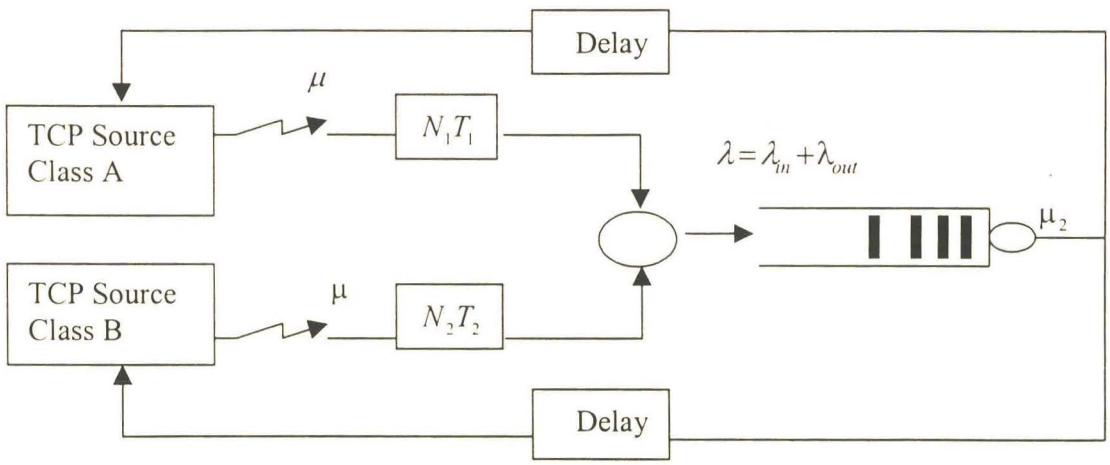


Figure 5.1. Analytical model

The number of connections sharing the bottleneck link, N_1, N_2, and so on is assumed to be constant. As in [26] all delays for each connection except for the service time and queuing at the bottleneck link are lumped together into a single propagation delay. Each of the connections TCP window control protocol is as explained in Section 3.2. Figure 5.1 summarizes our analytical model.

5.2.2 Differentiated Services

Packet loss due to congestion happens only in one congestion buffer of size B. The assured packets are forwarded into the low priority buffer employing a RIO scheme as in Section 2.3.2.2.1 with IN and OUT packets receiving different loss probabilities. The packets are lost due to overflow of the finite queue and due to the active queue management as done in RIO.

5.2.2.1 Packet marking

The packet is marked as IN or OUT depending on the sending rate of flow and its reservation rate. Let R be the reservation window, its size is found from equation 5.1.

Figure 5.2. TCP window marking

$$R = \frac{R_r}{k} * RTT \qquad (5.1)$$

where R_r is the reservation rate and k is the packet size and RTT is the round trip time. Every packet in the window is marked as IN when the window size is less than R and out when otherwise, i.e. the first R^{th} packets in a window are marked as IN and the rest are marked as OUT when the window size is greater than R (see figure 5.2).

5.2.2.2 Differentiated services model

The congestion buffer has IN and OUT packets coming in from the various TCP sources with rates λ_{in} and λ_{out} packets per second respectively and is served at a rate μ packets

per second. The DiffServ is implemented by a RIO mechanism with the parameters of the two RED algorithms chosen as shown below.

- The total queue length $avg-total$ was used in calculating the probabilities for both the IN and OUT packets
- The minimum and maximum thresholds for the IN and OUT packets $\min-in_{th}$, $\max-in_{th}$ and $\min-out_{th}$, $\max-out_{th}$ were chosen to be the same.
- The maximum dropping probabilities for IN and OUT packets $P_{\max-in}$ $P_{\max-out}$ were chosen as follows, $P_{\max-out} > P_{\max-in}$. Thus we achieve the discrimination between the IN and OUT packets.

5.2.2.3 *Analysis of the DiffServ Model*

From the description of the DiffServ model, it is apparent that the packets are dropped or discouraged from entering the queue and therefore we choose to model the RIO congestion buffer analytically as $M/D/1/K$ queue with state dependent (queue length dependent) arrival as described below.

Let the IN packets arrive according to a Poisson process with rate λ_{in}, and the OUT packets arrive according to a Poisson process with rate λ_{out} to a buffer of size B. The sum of Poisson random variables, $\lambda = \lambda_{in} + \lambda_{out}$ is a Poisson distribution. Let $P_a(n)$ be the probability that an arriving packet is accepted with n packets in the system, $P_a^{in}(n)$ be the probability that an arriving IN packet is accepted with n packets in the system and $P_a^{out}(n)$ be the probability that an arriving OUT packet is accepted with n packets in the system. The acceptance probabilities are given by

$$\begin{cases} P_a^{in}(n) = 1 \\ P_a^{out}(n) = 1 \end{cases} \quad n \le B/2 \tag{5.2}$$

$$\begin{cases} P_a^{in} = 1 - P_{\max-in}(2n-B)/B \\ P_a^{out} = 1 - P_{\max-out}(2n-B)/B \end{cases} \quad B/2 \le n \le B-1$$

Let $\lambda_a^{in}(n)$ be the arrival rate of an IN packet with n packets in the system and $\lambda_a^{out}(n)$ be the arrival rate of an OUT packet with n packets in the system. These state dependent arrivals are given by

$$\lambda_a^{in}(n) = \lambda_{in} P_a^{in} \tag{5.3}$$

$$\lambda_a^{out}(n) = \lambda_{out} P_a^{out}$$

The state dependent M/D/1/K is analyzed using embedded Markov chain approach [57].

5.2.2.3.1 *Departure Point Probabilities for an $M/G/1/K$*

Let X_n be the queue length after departure of customer n, the embedded stochastic process X_n of the number in the system at successive completion of service times has been shown to be Markovian [57]. Let A_n be the number of customers who arrive during the service time S^n of the n th customer and let $B(t)$ denote the cumulative distribution of the service times which are independent and identically distributed. The service times are independent of the previous service times and the length of the queue. The number in the system after the departure of the next customer is given by

$$X_{n+1} = \begin{cases} X_n - 1 + A_{n+1} & (X_n \geq 1) \\ A_{n+1} & (X_n = 0) \end{cases} \tag{5.4}$$

The single step transition matrix for the Markov chain is given by

$$\begin{aligned} \Pr\{X_{n+1} = j / X_n = i\} &= \Pr\{A_n \equiv j - i + 1\} \\ &= k_i \end{aligned} \tag{5.5}$$

where k_i is the probability that i packets have arrived during the service time of the n^{th} packet and is given by

$$k_i = \begin{cases} \int_0^{\infty} \frac{e^{-\lambda t}(\lambda t)^{j-i+1}}{(j-i+1)!} dB(t) & (j \geq i-1, i \geq 1) \\ 0 & (j < i-1, i \geq 1) \end{cases} \tag{5.6}$$

For an $M/D/1/K$ the service time T is constant. In our analysis we use a mean input rate λ_k^{av} and our input process becomes a filtered Poisson process with this mean. The

justification for using the mean is as given in *Appendix 3*. The probability of k_i is now given by

$$k_i = \begin{cases} \dfrac{e^{-\lambda_i^{av}(\lambda_i^{av}T)}}{(j-i+1)!} & (j \geq i-1, i \geq 1) \\ 0 & (j < i-1, i \geq 1) \end{cases} \tag{5.7}$$

where λ_k^{av} is given by equation (5.8) and λ_l is the Poisson arrival rate.

$$\lambda_k^{av} = \frac{\sum_{l=i}^{i+k} \lambda_l}{k} \tag{5.8}$$

The single step transition matrix is shown in equation 5.9. Due to the system's finite size during a service period, the expected joined arrivals are conditioned on the system size and hence the Pollaczek-Khintchine formula doesn't hold. The steady state probabilities are solved by solving the stationary equations.

$$P = \begin{bmatrix} k_0 & k_1 & k_2 & \cdots & 1-\sum_{n=0}^{K-2} k_n \\ k_0 & k_1 & k_2 & \cdots & 1-\sum_{n=0}^{K-2} k_n \\ 0 & k_0 & k_1 & \cdots & 1-\sum_{n=0}^{K-3} k_n \\ 0 & 0 & k_0 & \cdots & 1-\sum_{n=0}^{K-4} k_n \\ 0 & 0 & 0 & \cdots & 1-k_0 \end{bmatrix} \tag{5.9}$$

From the transition probability matrix we find our steady state probability matrix π_n, which is the probability that a departing or arriving packet leaves n packets in the queue. We solve the matrix for steady state probabilities using the Laplace Stieljies method [58]. With the steady state probabilities the IN and OUT packet blocking probabilities with n packets in the system $P_{cl}^{in}(n)$ and $P_{cl}^{out}(n)$ are then found from the following equations

$$P_{cl}^{in}(n) = 1 - \sum_{n=o}^{B} P_{cl}^{in}(n)\pi_n \tag{5.10}$$

$$P_d^{out}(n) = 1 - \sum_{n=0}^{B} P_a^{out}(n)\pi_n$$

5.2.3 Wireless Channel Model

The wireless channel is modeled by the Gilbert Elliott Channel as explained in Section 4.2.2.1. It is modeled by a simple two state Markov chain with transition matrix M_c given by equation 4.7.

5.2.4 General Approach

5.2.4.1 *Overall throughput*

The higher layer is modeled by ON/OFF sources that generate TCP packets of fixed sizes in the ON state. The number of the packets is Poisson distributed with a mean of n .
The overall throughput, T is given by

$$T = \sum_n f(n)P(n) \tag{5.11}$$

where $P(n)$ is the Poisson distribution that generates the number of packets and $f(n)$ is the throughput conditioned on the number of packets generated. When given the packets the TCP layer evolves in cycles, with cycle C_i taking a time of T_i seconds as shown in figure 5.3. The n packets can be transmitted in about m cycles. A TCP cycle begins after packet loss detection with either slow start or congestion avoidance and ends with the successful conclusion of fast recovery mechanism or on the basis of a timeout. The cycle's window evolution can be constrained by the factors shown below.

- At the beginning of TCP flow establishment the receiver advertises a maximum buffer size W_{rec}. This fixes the upper limit on the maximum window size.
- The link fixes the maximum window size. This is the maximum window size W_m allowed on the constraint of the sum of bandwidth delay product and the buffer size. This has been determined in [26] as $W_m = \mu T + 1$.

The cycle terminates when the window W is given by

$$W = \min\{W_{rec}, W_m, W_d\} \tag{5.12}$$

where W_d is the window when a packet drops due to congestion or the wireless channel losses. Let Q_c be the number of packets transferred during a cycle and T_c the time taken, and Q_t be the summation of the actual packets transmitted in all the m cycles and T_t the total time. The actual throughput can be looked at as the ratio of the packets transmitted and the time it takes to transmit them. It is given by

$$T = \sum_n \frac{Q_t}{T_t} P(n) \tag{5.13}$$

where Q_t and T_t are given by

$$Q_t = \sum_{c=1}^{m} Q_c \tag{5.14}$$

$$T_t = \sum_{c=1}^{m} T_c$$

The values of the packets transmitted in a cycle and the total time taken are calculated in the next section.

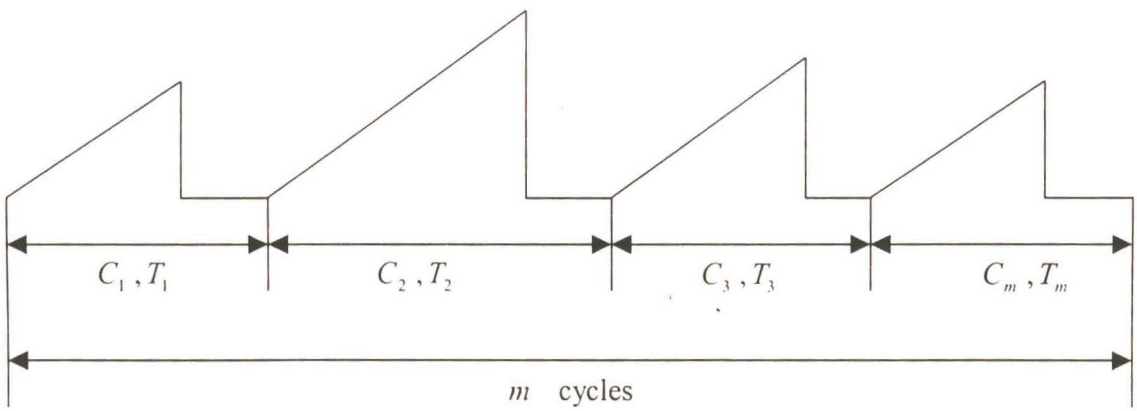


Figure 5.3. TCP cycle evolution

5.2.4.2 TCP Window Evolution in the Cycle

For the TCP window evolution in the cycle we follow the analysis given in Section 3.3.2. We get the expectation of the packets transmitted in a cycle Q_c from equation (3.8) and the cycle time from equation (3.10). However, we have the following differences in determining the parameters.

- The window evolution is conditioned on the number of packets generated by the application layer.
- We have two types of packets in the system, the IN packets and the OUT packets and therefore our window evolves as in figure 5.4.

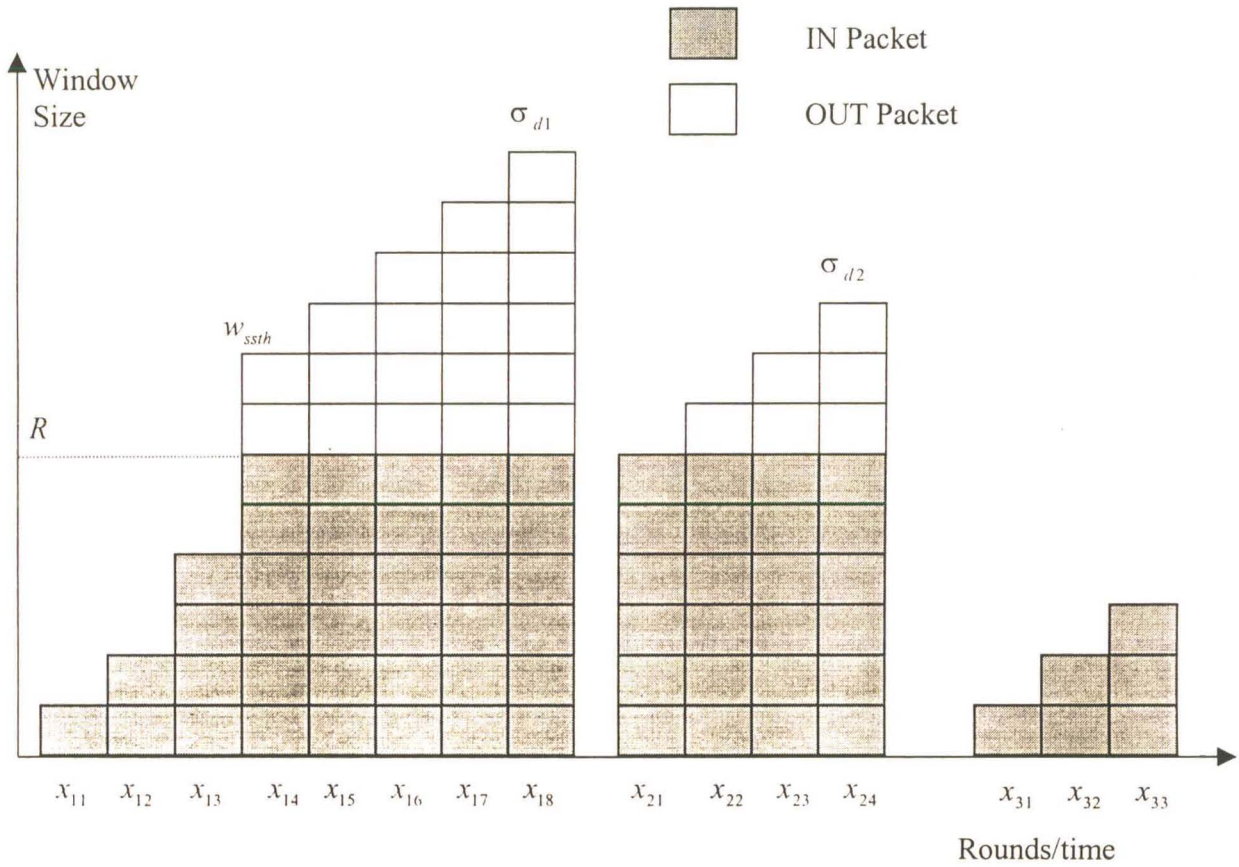


Figure 5.4 TCP window evolution with IN/OUT

The explanation of figure 5.4 is analogous to figure 3.1 Chapter 3. The differences arise with the addition of the marking rate R and the IN and OUT packets as indicated. Because of the changes in our system the calculation of the following, loss window probability distribution $P(\sigma)$, the conditional round probability distribution $P(x_1/\sigma)$, the conditional packet count calculation $(Q_1/x_1, x_2, \sigma)$ and the timeout probability calculation τ_i also change. We derive them in the following sections.

5.2.4.3 Loss window probability calculation

The steady state loss window probability $P(\sigma)$, $\sigma = 1, 2, \ldots, W_m$ is calculated the same way as in Section 3.3.2.1. It is found from the transition probability matrix given by 3.12 and is repeated here for clarity.

$$P(U_{p+1} = \eta / U_p = \sigma) = \begin{cases} D_{s\sigma} S_s(\eta) F_s(\eta) & \text{for } 1 \le \eta \le \eta_{ss} - 1 \\ D_{s\sigma} S_c^s(\eta) F_c(\eta) & \text{for } \eta \ge \eta_{ss} \\ D_{c\sigma} S_c^c(\eta) F_c(\eta) & \text{for } \eta \ge \eta_{ss} \end{cases} \quad (5.15)$$

where η_{ss} is the slow start threshold. The differences arise in the calculation of the terms of the equation.

5.2.4.3.1 Calculation of $F_s(\eta)$

As in Section 4.3.2, η_k is the drop window size during cycle of interest in round k. The position of the last successful packet in the round is n_s i.e. it's the $(n_s + 1)$th packet in the round, the position of dropped packet $n_d = n_s + 1$. The events that lead to the first packet and the next being dropped are as follows.

- The event that a packet dropped given that the previous one was delivered and is not the first occurs when the channel changes from good to bad or when the channel changes from good to good and packet is lost due to congestion. Its probability is given by

$$\begin{cases} b = \alpha + (1-\alpha) P_{in} & \text{if } \eta \le R, n_{ss} > 0 \\ c = \alpha + (1-\alpha) P_{out} & \text{if } \eta > R, n_{ss} > 0 \end{cases} \quad (5.16)$$

  where R is the marking rate. Clearly b is the probability of an IN packet while c is the probability of an OUT packet.

- The event that the first packet is dropped occurs when you find the channel in bad state and it remains there (a packet is dropped) or when you find the channel in bad state and it changes to good state (packet succeeds) but is lost due to congestion or find channel in good state and it changes to bad state (packet is dropped) or find

packet in good state and remains there (packet succeeds) but is dropped due to congestion. Note that the first packet is an IN packet. The event probability a is

$$a = \{\pi_B(1-\beta)+\pi_G\alpha+\pi_G(1-\alpha)P_{in}+\pi_B\beta P_{in} \quad \text{if } n_s = 0 \tag{5.17}$$

We know the position of the dropped packet in a round $n_d = n_s + 1$ from the loss window, therefore the previous packets were delivered and hence the probability for a packet loss for a window in slow start is given by

$$F_s(\eta) = \begin{cases} a & \text{if } n_d = 1 \\ b & \text{if } 1 < n_d \le R \\ c & \text{if } 1 < n_d > R \end{cases} \tag{5.18}$$

5.2.4.3.2 Calculation of $S_s^s(\eta)$

As in Section 4.3.3, with η_k the loss window and therefore k the loss round, with $(k-2)$ the number of full successful slow start rounds, $S_s^s(\eta)$ is found from 4.18, as $S_s^s(\eta) = P_{all} * S_{lw}$, where $P_{all} = \prod_{j=1}^{k-2} P_j$. The difference arises in the calculation of P_j.

Let X be the probability that the first packet in the round succeeds, Y the probability that the next packet succeeds if it is an IN packet and Z the probability that the next packet succeeds if it is an OUT packet. All the packets in a round succeed if the first packet succeeds and all the next/subsequent packets in the round succeed. The first packet succeeds if it finds the channel in a good state and succeeds and it is not lost due to congestion or if it finds the channel in a bad state and succeeds and is not lost due to congestion. The next packet succeeds if the channel changes from good state to good state and the packet is not lost due to congestion. These probabilities are given by

$$X = \pi_G(1-\alpha)(1-P_{in})+\pi_B\beta(1-P_{in}) \tag{5.19}$$

$$Y = (1-\alpha)(1-P_{in}) \quad \text{if the next is an IN packet}$$

$$Z = (1-\alpha)(1-P_{out}) \quad \text{if the next is an OUT packet}$$

We continue with our notation and we denote η_j as the window size of round j where $0 < j \le k-2$. The probability P_j is given by

$$P_j = \begin{cases} XY^{\eta_j - 1} & \text{if } \eta_j < R \\ XY^{R-1}Z^{\eta_j - R} & \text{otherwise} \end{cases} \tag{5.20}$$

from which we find

$$P_{all} = \begin{cases} X^{k-2}\left[\prod_{j=1}^{r} Y^{\eta_j - 1}\right]\left[\prod_{k=r}^{k-2} Y^{(R-1)}Z^{\eta_j - R}\right] & \text{if } r \le k-2 \\ X^{k-2}\left[\prod_{j=1}^{k-2} Y^{\eta_j - 1}\right] & \text{otherwise} \end{cases} \tag{5.21}$$

where $r = \log_2 R + 1$ is the round where the window reaches the marking rate. The probability of packets succeeding in this round is given by

$$S_{hw}(\eta) = \begin{cases} XY^{n_s - 1} & \text{if } n_s \le R \\ XY^{R-1}Z^{n_s - R} & \text{if } n_s > R \end{cases} \tag{5.22}$$

5.2.4.3.3 Calculation of $S_c^c(\eta)$, $F_c(\eta)$ and $S_s^c(\eta)$

Similarly we follow Section 4.3.4 and combine with the above section. We note that the window has just been shifted and from the previous section the probability of packet success and failure are given by

$$S_c^c(\eta_k) = \begin{cases} X^{k-2}\left[\prod_{j=1}^{r} Y^{\eta_j - 1}\right]\left[\prod_{k=r}^{k-2} Y^{(R-1)}Z^{\eta_j - R}\right] & \text{if } r \le k-2 \\ X^{k-2}\left[\prod_{j=1}^{k-2} Y^{\eta_j - 1}\right] & \text{otherwise} \end{cases} \tag{5.23}$$

$$F_c(\eta_k) = \begin{cases} 1 - XY^{\eta_k - 1} & \text{if } \eta_k < R \\ 1 - XY^{R-1}Z^{\eta_k - R} & \text{if } \eta_k > R \\ 1 & \text{if } \eta_k = W_m \text{ since a packet is surely lost} \end{cases} \tag{5.24}$$

The probability of packets success from slow start to congestion avoidance $S_s^c(\eta)$ as described in Section 4.3.5 is a combination of the above and is given by

$$S_s^c(\eta) = S_s^s(\eta) * S_c^c(\eta) \tag{5.25}$$

5.2.4.4 Timeout probability calculation

The timeout probability is calculated as done in Section 3.3.2.5. The only difference arises in calculating the probability of a packet loss which is shown in this section. A packet drop event occurs when the packet finds the channel in bad state or finds the channel in good state and packet is dropped due to congestion. The steady state channel probabilities π_G and π_B are given by equations (4.10). The probability that an IN packet is lost, a, and the probability that an OUT packet is lost, b, are given by

$$a = \pi_B + \pi_G P_{in} \qquad (5.26)$$

$$b = \pi_B + \pi_G P_{out}$$

Let i represent lost packets, P_i be the probability that i packets are lost. The event that i packets are lost implies that j IN packets are lost and $i-j$ OUT packets are lost. Its probability is given by

$$P_i = \begin{cases} \sum_{j=0}^{i} \left[\binom{R}{j} a^j (1-a)^{R-j} \right] \left[\binom{\eta - R}{i-j} b^{i-j} (1-b)^{\sigma - R - i + j} \right] & \text{if } \eta > \mathrm{R} \\ \sum_{j=0}^{i} \left[\binom{\eta}{j} a^j (1-a)^{\eta - j} \right] & \text{if } \eta \le \mathrm{R} \end{cases} \qquad (5.27)$$

where η is the loss window and R is the marking rate. With P_i, we further calculate the timeout probability as in Section 3.3.2.5.

5.2.4.5 Packet count calculation

Let θ_k be the number of packets in round k and j be the slow start round, η_j be the first slow start window, R the marking rate, σ the loss window. We want to find the number of packets delivered Q_1 in the first phase, i.e. until a loss occurs given the number of rounds m and the loss window σ. It is a sum of the IN and OUT packets as given by

$$E(Q_1 / x_1 = m, \sigma) = E(N_{in}) + E(N_{out}) \qquad (5.28)$$

Let $r = (\log_2 R) + 1$ be the round where the window reaches the reservation rate R. The sum of the IN and OUT packets is given by

$$E(N_{in}) = \sum_{i=1}^{r} \theta_i + \sum_{i=r}^{m} \theta_r \tag{5.29}$$

$$E(N_{out}) = \sum_{i=r}^{m} (\theta_i - \theta_r)$$

where the number of packets in round k, θ_k is given by

$$\theta_k = \begin{cases} 2^{k-1} & \text{if } k \le j \\ \theta_{k-1} + 1 & \text{if } j < k \le m \end{cases} \tag{5.30}$$

and $J = (\log_2 \eta_j) + 1$ is the slow start threshold round with the slow start threshold being given by half the previous loss window, $\eta_j = \sigma/2$. This is just counting the number of packets and is clearly depicted when you look at figure 5.4.

Note: The calculation of the Expected round and any other parameter like $D_{s\sigma}$ and $D_{c\sigma}$ is done as in Section 4.3.

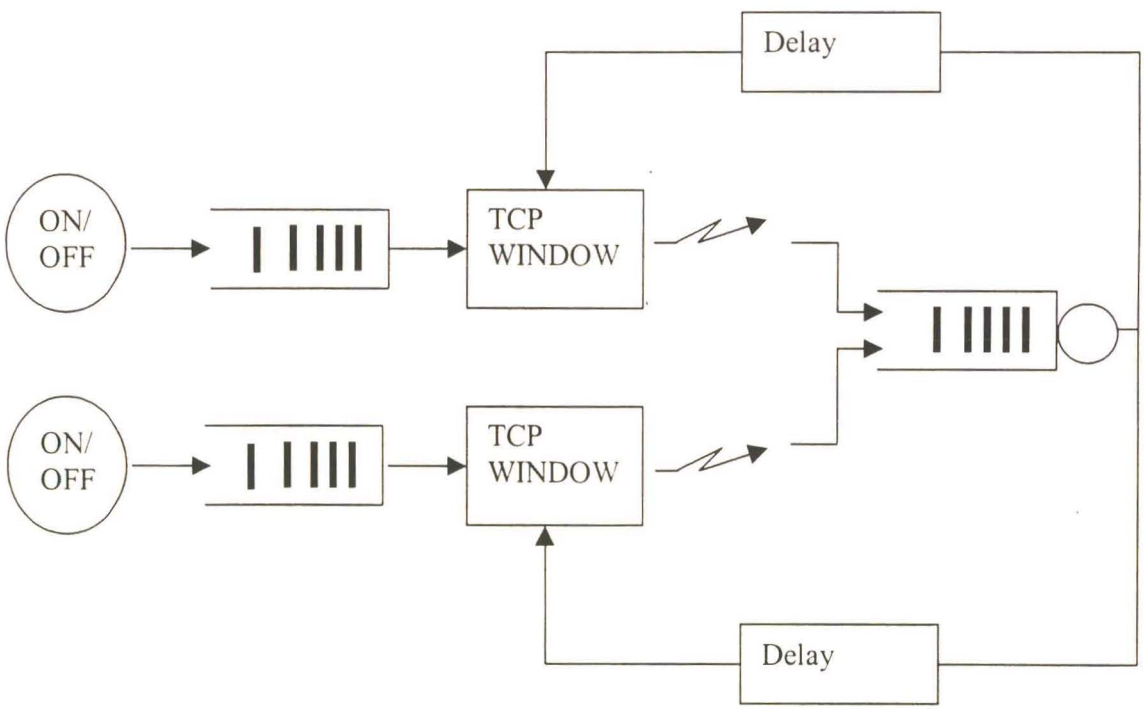


Figure 5.5. Simulation model

5.3 SIMULATION MODEL

To validate the analysis in Section 5.2 we have our simulation model as shown in figure 5.5. We used sources with different classes, where a class is distinguished by the marking rate. The sources generate traffic according to an ON/OFF model and then feeds them into a buffer. The TCP window evolution is modeled according to the basic window algorithm as in Section 3.2. However, we impose a condition to match the analysis. The condition is that when the window reaches its maximum size a packet is surely dropped. It is then fed to a wireless channel which is modeled by the Gilbert Elliott channel model as in Section 4.2. It then has a finite congestion buffer which models all the network routers. There is also a delay which models the subsequent time taken by the packet in the network. We make an assumption that the acknowledgements are not lost, this is logical owing to their small sizes. The simulation was then done using a developed discrete event simulator in C++ programming language and its algorithm is as given in Appendix 4.

5.4 LIMITATIONS OF ANALYSING AND SIMULATING TCP OVER WIRELESS WITH DIFFERENTIATED SERVICE

The analysis of the TCP Protocols has various setbacks and they are outlined as below

- Protocol complexity: Due to its work the TCP protocol is a complex protocol, it has very many parameters to take care of. These include the timers, the window parameters and many more. The incorporation of all these parameters in the analysis can lead to a complex analysis which is mathematically not tractable. Another factor to consider is the availability of many versions of the TCP protocol, each coming with additional complexity.
- Computational complexity: This problem has also been sighted in [59]. Equation (3.8) and (5.15) are iterative in nature. We iterate until we reach a steady state. In this iteration, getting the steady state probabilities of the queues is done analytically which involves more inner iterations. The iterations have to go on until the throughput converges to a steady state value. The iterations require an enormous

amount of CPU time and this increases as the size of parameters like the window size and buffer size increase.

- Limited window: We have two types of limitations on the maximum window; the first is due to the limit imposed by the computation complexity as discussed above. The second is in choosing our events where we look for the probability of the window reaching a maximum. We don't model the probability of it staying at a particular window size.
- Limited queue size: The queue size is limited by two factors. One is the computation complexity as indicated above. The other factor is the fact that for there to be a considerable differentiated service we need some form of congestion in the network and hence drop packets, very large queues would not be of any benefit.
- Rapid degradation of throughput with slight congestion: The other problem is the rapid degradation of throughput with slight congestion. This makes quantifying throughput at a considerable amount of congestion slightly difficult.
- The simulation model has only one problem of requiring enormous amount of memory while the number of sources increases. This led us to fix the number of sources and introduce congestion by reducing the wired link rate.

Table 5.1. Simulation/Analytical Parameters

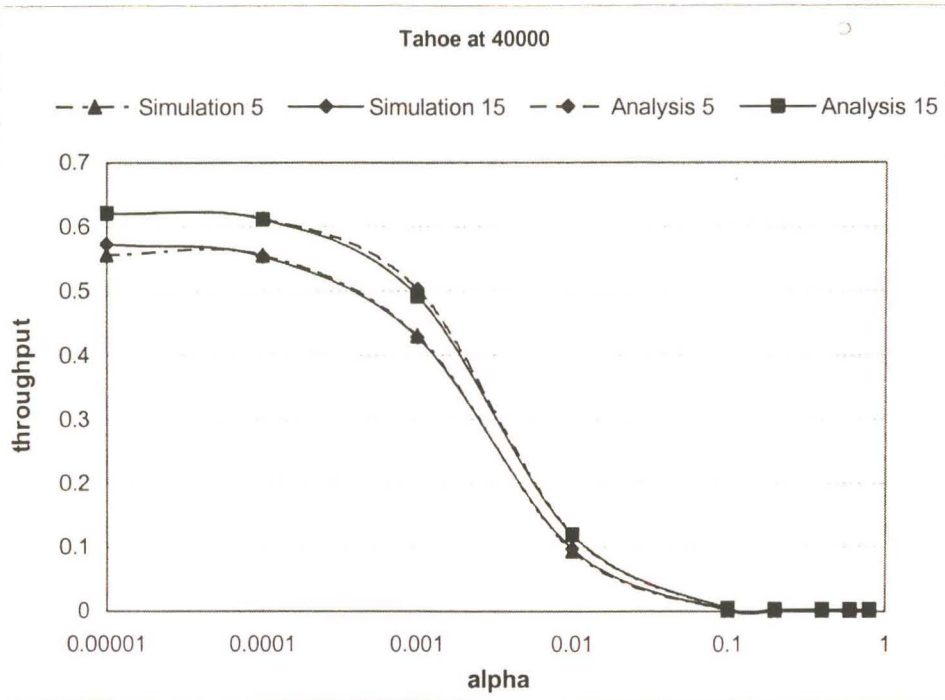
Parameter	Value
Beta	0.1
Wireless Link	2000pkts/s
Retransmission time	100ms
TCP Timeout value	500ms
TCP Ack threshold	3
TCP Maximum Window	20 Pkts
No. of Sources	20
Pmax_In	0.1
Pmax_Out	0.9
Buffer size	50

5.5 RESULTS AND DISCUSSION

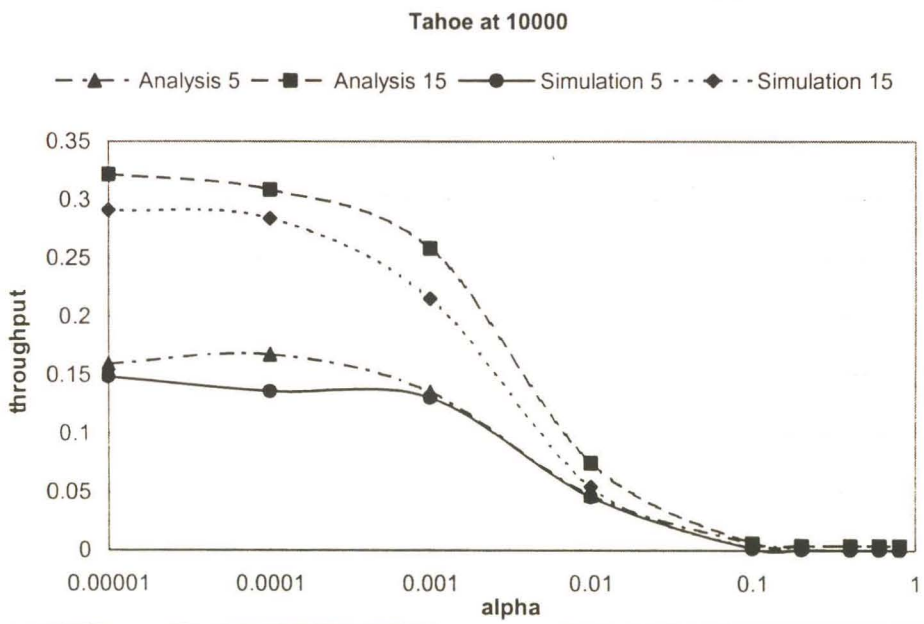
In the analysis and discussion of our models the parameters were kept as indicated in table 5.1 unless otherwise specified. In our analytical and simulation results we plot the throughput against alpha which is an indication of the packet loss probability as shown in Section 4.2.2.1. The throughput is actually normalised to the wireless link bandwidth. There are various ways of introducing congestion in our model, one is by increasing the number of sources, the other is by reducing the buffer size. The last method is by reducing the fixed links mean rate. Due to the limitations in the analysis and simulation we choose to introduce congestion by reducing the fixed link rate and we refer to this fixed link rates as congestion levels in the thesis. We analyse and simulate twenty sources on the network, the sources are grouped into several classes where the classes are distinguished by the different marking rates. The rates are given in terms of the number of packets reserved in the window. However their values in bits per second can be easily computed from equation (5.1).

5.5.1 Results and Discussion of Tahoe

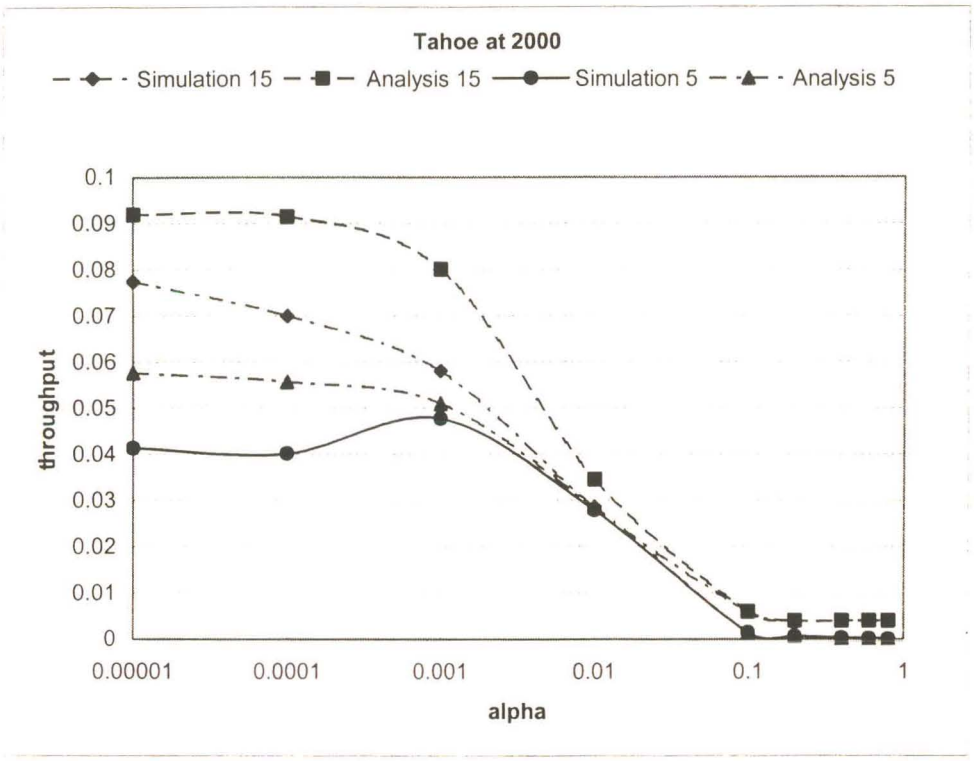
Figures 5.6 a-c shows the analytical calculations of our model and the simulation results of TCP Tahoe. In both the throughput degrades with increasing probability but different traffic classes are affected differently at different congestion levels. At a congestion level of 40000bps in figure 5.6a, we find that there is basically no congestion. The TCP sources behave just as they behave on a plain wireless channel with their throughput reducing with increasing drop probability. There is no difference between the different traffic classes. We noticed that the queue didn't build up in this case and so the RIO algorithm was not dropping any packets. It should be noted that even with no congestion probability and at a very low wireless loss probability, as low as 1E-5, the throughput of TCP Tahoe is not 1. This is attributed to the fact that when the window reaches its maximum value it drops a packet. The dropping of a packet plus the time the window takes to reach the maximum value again is a waste of bandwidth and thus the maximum throughput cannot be achieved. The maximum normalised throughput is about 0.6.



a) Wired channel rate 40000



b) Wired channel rate 10000



c) Wired channel rate 2000

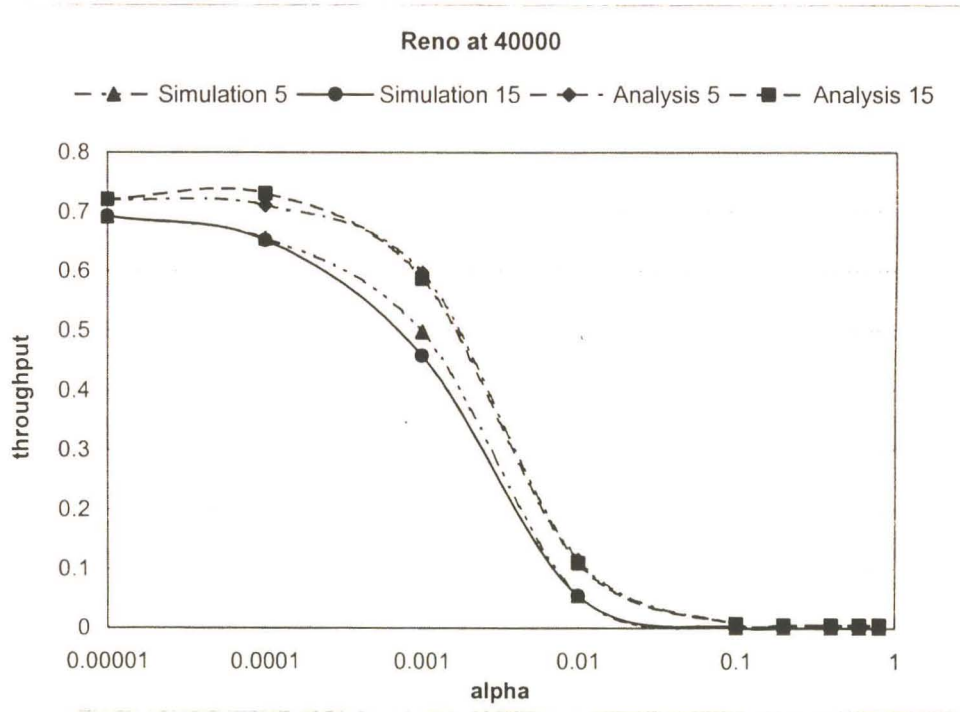
Figure 5.6. Tahoe simulation and analysis

As the congestion level reduces to 1000bps, the maximum throughput at the lowest alpha for both traffic classes has reduced. An interesting phenomenon is taking place. There is a marked difference in the throughput of both traffic types. The queue size builds up and the RIO algorithm drops packets differentially. This leads to the different traffic classes achieving different rates. As predicted the source with the highest marking rate, Tahoe15 achieves the highest throughput. The source with the largest number of IN packets achieves higher throughput. Hence at this rate it is possible to give relative differentiated services. Note that the order of throughput and rates is the same for both analysis and simulation. However, the actual throughput may not be the same. This is currently a problem in that DiffServ is not good at providing absolute differentiated services, but can provide relative differentiated services. This clearly indicates that the throughput reduces with a reduction in congestion level. The throughput also reduces with a reduction in marking rate at a high congestion level. In all the results we see that the analytical values

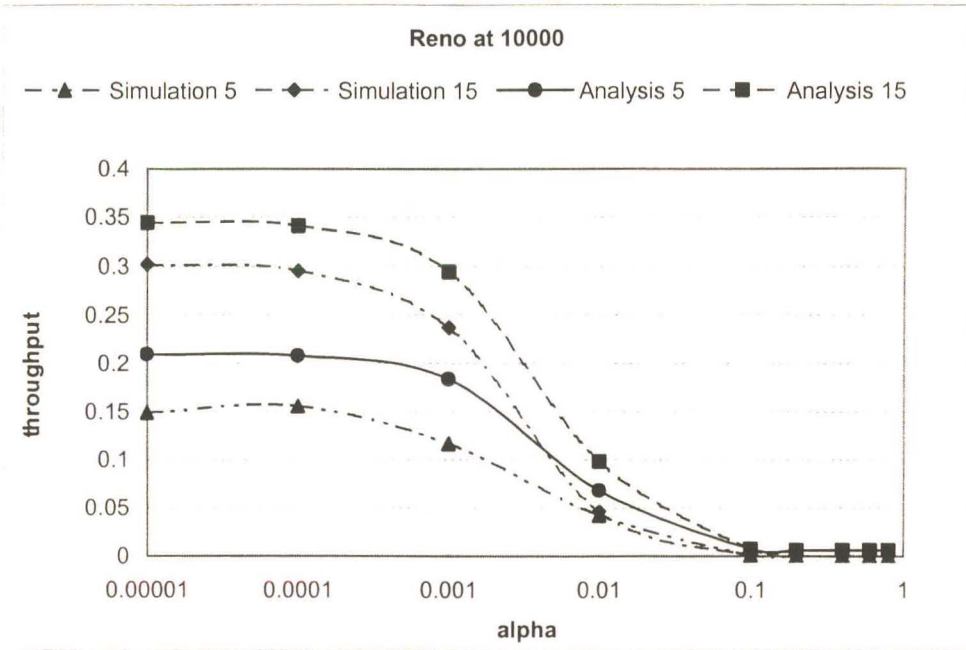
are slightly higher than simulation values. We will look at the reason why later in the section.

5.5.2 Results and Discussion of Reno

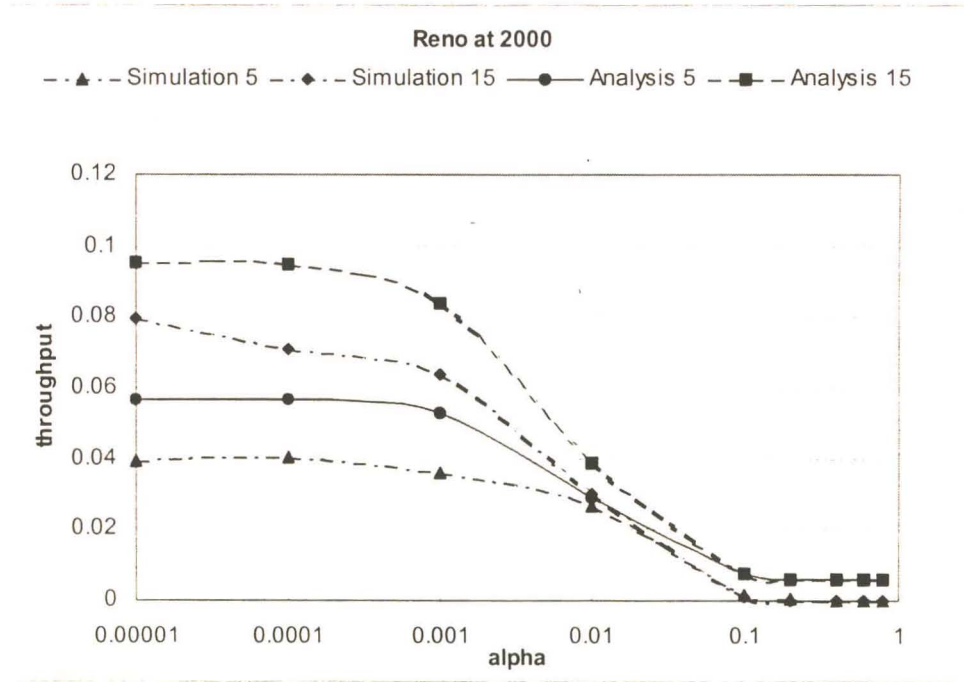
The analytical and simulation results for twenty Reno sources of four different classes with reducing congestion level are shown in figure 5.7. They are analogous to the results of Tahoe. It is seen that the throughput degrades with increasing packet error probability. At the lowest congestion level, both classes behave the same since there is no sizeable build up of the queue. As the queue builds up with increasing congestion levels there is differentiated services and the classes receive different throughputs, with the class with the highest marking rate,15, getting the largest share of the bandwidth. As is the case of Tahoe the analytical results are slightly higher than the simulation results.



a) Wired channel rate 40000



b) Wired channel rate 10000

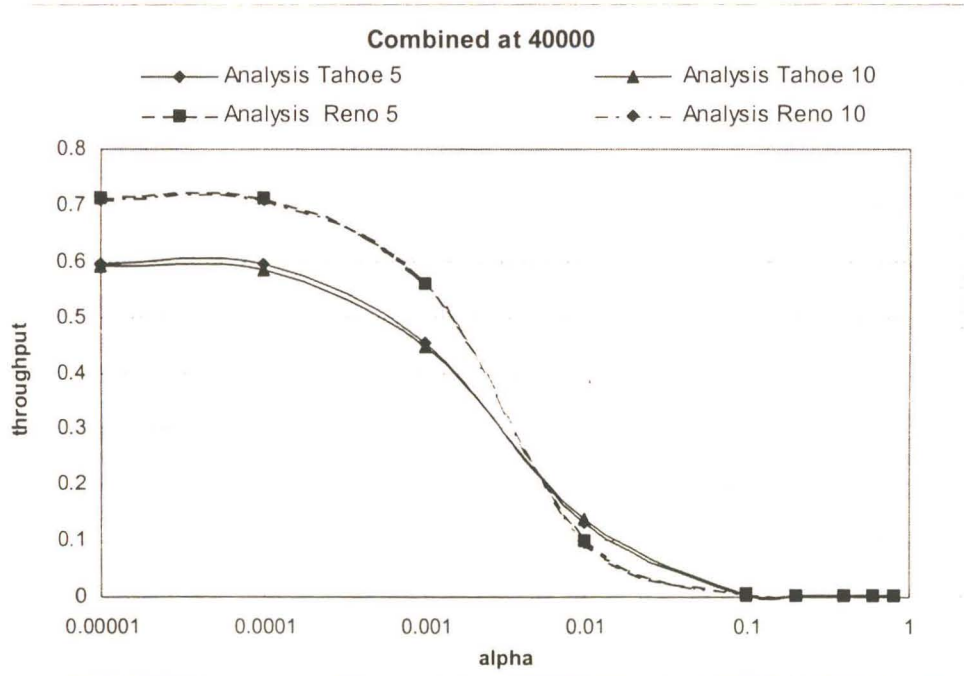


c) Wired channel rate 2000

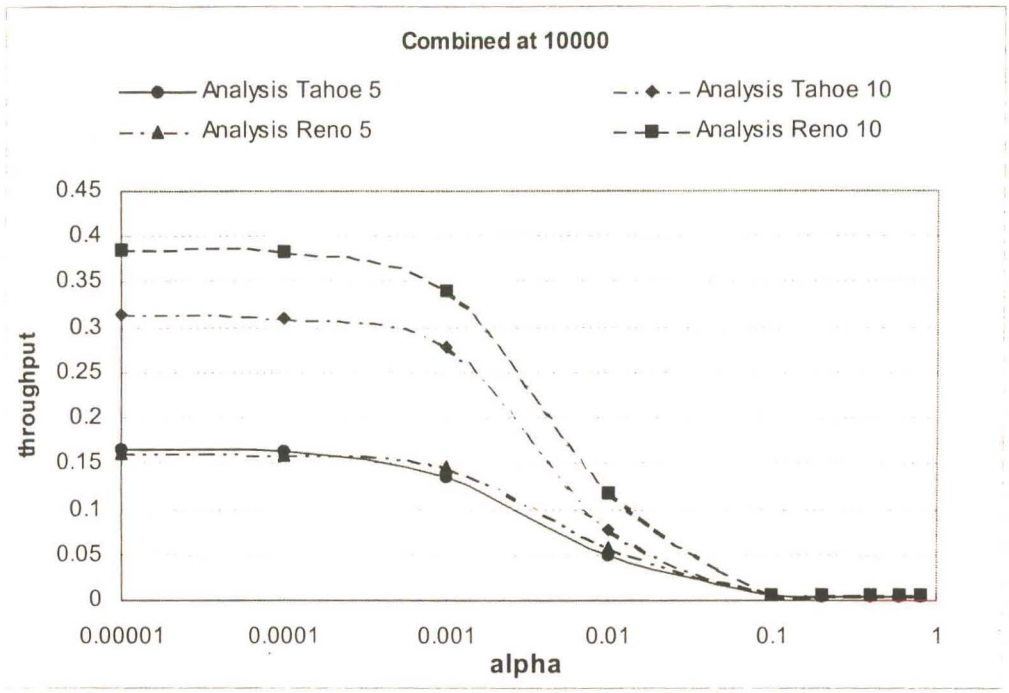
Figure 5.7. Reno Simulation and Analysis

5.5.3 Results and Discussion of Combined Tahoe and Reno

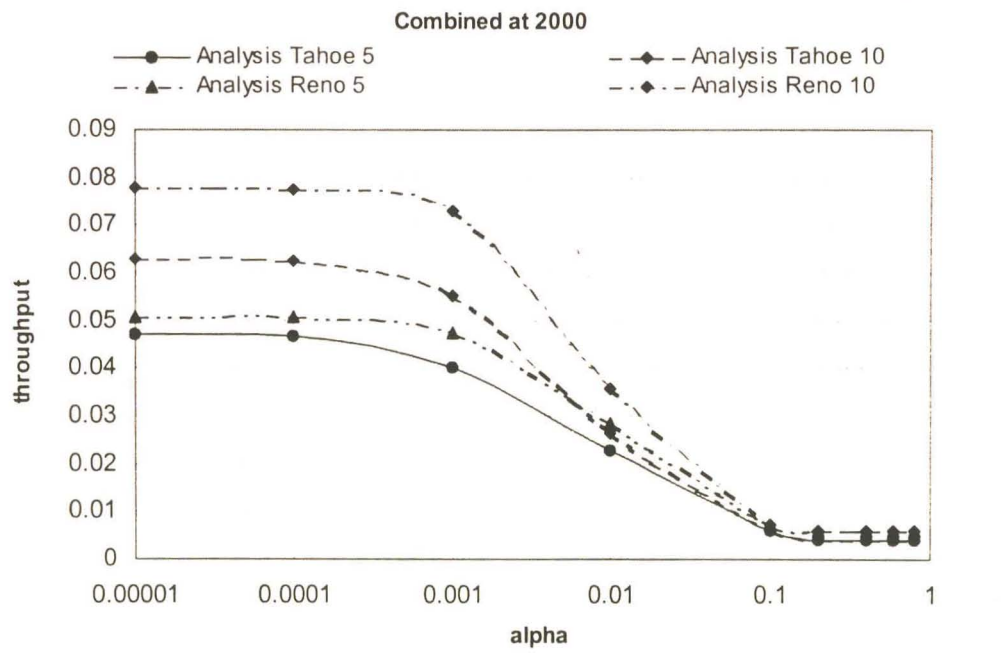
We consider a network with both Tahoe and Reno. Using our analytical model we plot the results of the throughput against the error probability as in figure 5.8a-c. The results exhibit the same behavior as discussed for both cases. The results at congestion level of 40000 bps shows both Reno sources performing better than the Tahoe sources at a higher value of alpha. As the value of alpha starts getting higher than 1E-3, Tahoe starts performing better than Reno. This is because Reno's first retransmit procedure does not succeed. It later resorts to timeouts for loss recovery having wasted time in first retransmit. As the congestion rate increases to level 10000, there is interclass differentiation. The sources with a marking rate of 10 achieve higher throughput than those with a marking rate of 5. At the highest congestion level there is negligible throughput. This is due to the many losses and the numerous timeouts.



a) Wired channel rate 40000



b) Wired channel rate 10000

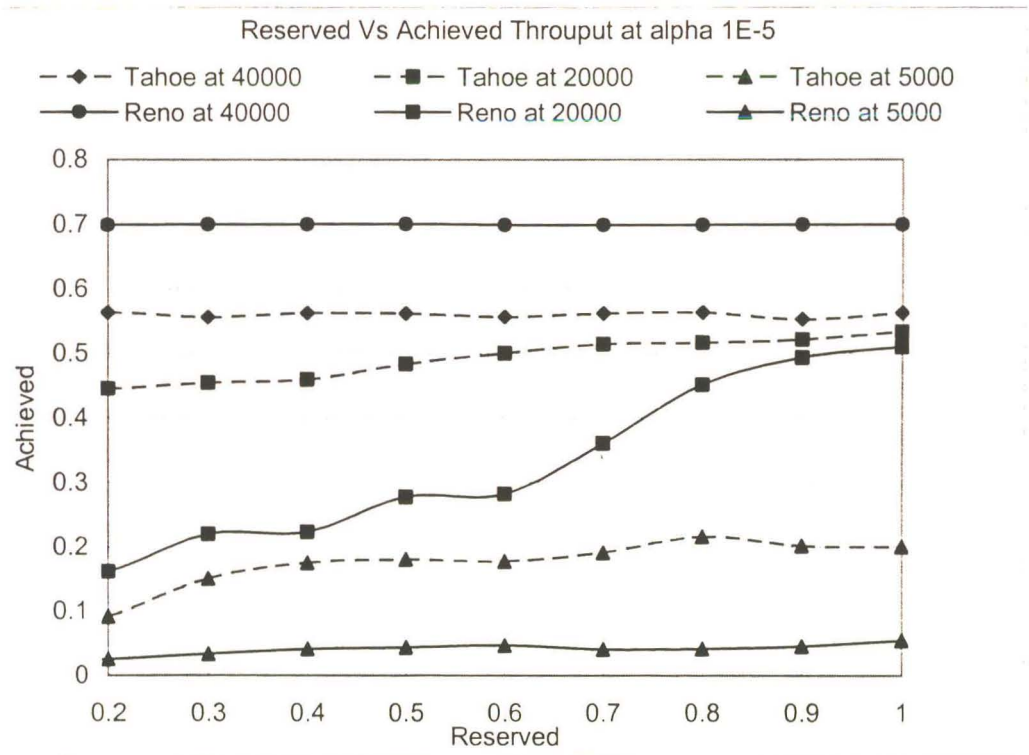


c) Wired channel rate 2000

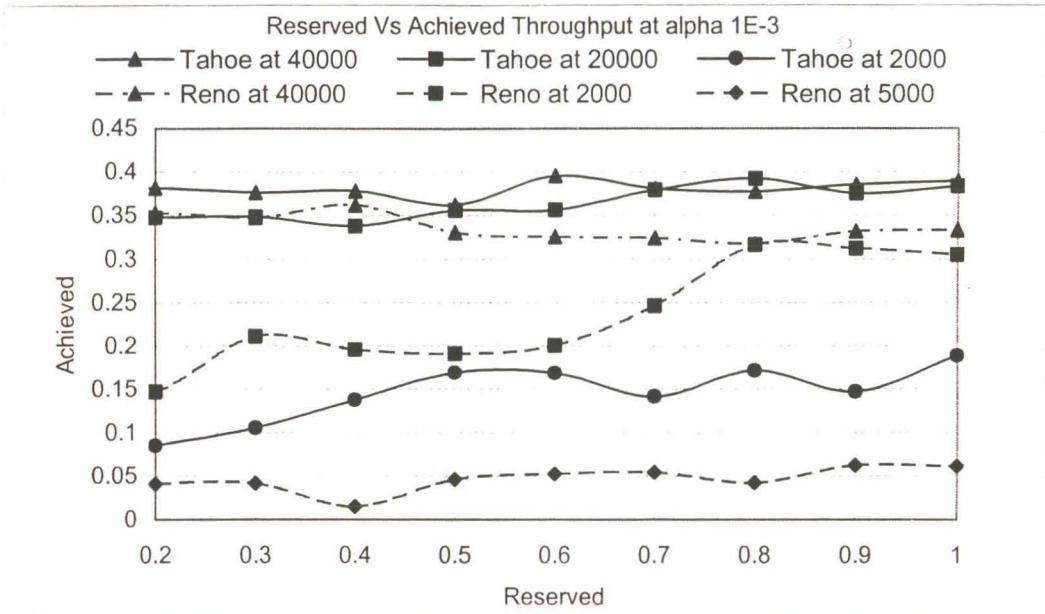
Figure 5.8. Combined results, Tahoe and Reno

5.5.4 Results and Discussion of achieved and Reserved Rate

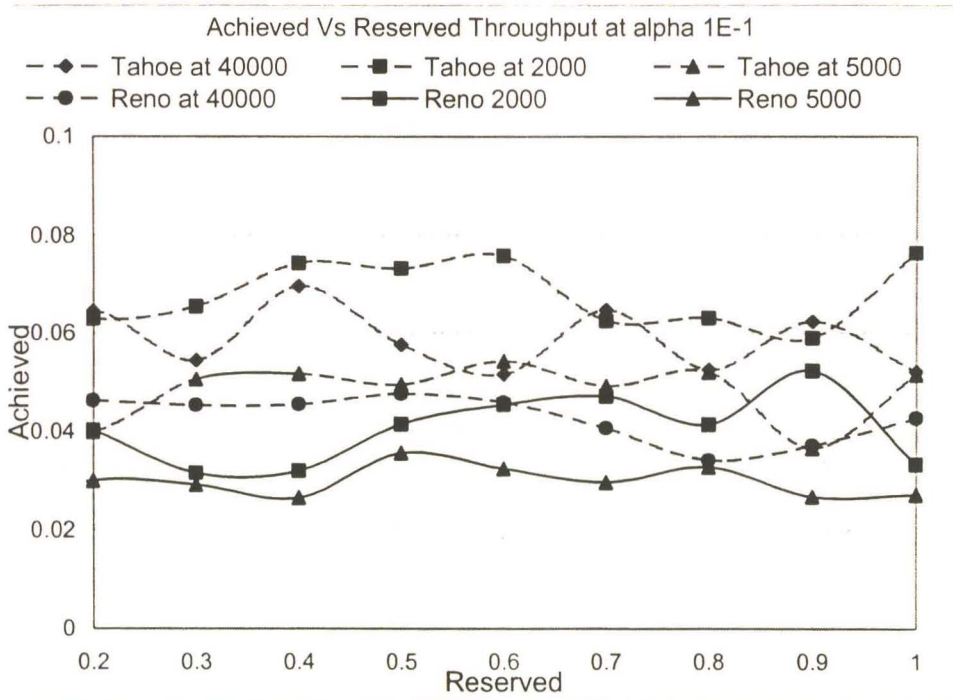
As in [56] and [34] the aim of DiffServ is to see if you can achieve the reserved bandwidth on the network. In this work they consider how to achieve the reserved bandwidth and at what error probability. The model considered in the work is not on a wireless channel and the IN and OUT packet are lost with a certain constant probability. This section answers the two questions. Firstly, do we achieve the bandwidth we have reserved and at what congestion level. Secondly, how is the excess bandwidth shared between users of different classes in the network. This question has also been dealt with in [60]. We used our model with twenty sources but with different reserved rates and congestion levels. The results were plotted in figure 5.9 at different values of alpha and congestion levels.



a) Alpha 1E-5



b) Alpha 1E-3



c) Alpha 1E-1

Figure 5.9 Results of reserved Vs achieved throughput

The graphs were plotted at the error value (alpha) of 1E-5, 1E-3, and 1E-1. The reserved and achieved rate are normalised to the wireless link bandwidth. The first point worth noting is that the maximum throughput that can be achieved with no error for Tahoe is about 0.6 and Reno is 0.7. Therefore a reservation of above this value is futile. At a lower error rate as in figure 5.10a, Both Tahoe and Reno at a congestion level of 40000 achieve their maximum rate since there are no errors. Tahoe at 20000 manages to achieve the reserved rate up to a value of 0.5 beyond which even though the achieved rate is increasing it is not as the reserved. Reno at 20000 does not however achieve its reserved rate even though the overall achieved rate increases as the level of reservation increases. The remaining sources do not achieve their rates; in fact they achieve an insignificant amount. This is because of the high congestion errors. At a lower wireless error rate of 1E-3 as in figure 5.10b, Tahoe at 40000 and 2000, with Reno at 40000 manage to achieve their reserved rates up to the rate of 0.4 from where they don't. At the lowest value of alpha, figure 5.10c, all the protocols fail to achieve the reserved rate and in fact their achieved rate is insignificant. The second question is what share of the reserved bandwidth do the traffic classes get. Figure 5.10a, gives us the results when we have excess bandwidth and with low losses. We see that for both Tahoe and Reno at 40000 the reserved vs. achieved graph is almost constant. It means they achieve the same throughput irrespective of the marking rate. From this we can clearly see that as far as the excess bandwidth is concerned the sources with lower marking rate get more of the excess bandwidth than the sources with the higher marking rate. The behavior is similar for the following graphs with reasonable throughput values.

Note:

- The loss rate in the simulation model was approximated as in [28] by taking the ratio of the packet lost to the packet transmitted. In the analysis the loss probabilities for the IN and OUT packets was averaged for all the iterations the system went through. When we compared the two we found that the loss probability of the analysis were slightly lower than the loss probability of the simulation. The reason is probably due to the approximation of the arrival rate in the buffer.

- The arrival rate in the queues was assumed to be Poisson. The Internet applications arrival process exhibits a long range behavior and could be better modelled by heavy tailed distributions like the Pareto distribution or the log extreme distributions than by a Poisson distribution. However from [61] it has been noted that a superposition of many ON/OFF sources can approximate a Poisson distribution. In both the analysis and simulation we used twenty ON/OFF models and hence we are justified to use the Poisson assumption.

5.6 SUMMARY

TCP over Wireless with Differentiated Services is a new area of research and not much work has been done. We have developed an analytical framework for analyzing TCP over wireless with Differentiated Services. Our analytical framework has the following merits.

- It can be used to model many sources in the network. The few analytical formulas developed, Section 3.3.3.1 can only model one source on the network.
- The analytical model is adaptive e.g. it can be used to model plain TCP over wireless.
- The analytical model can be extended to any version of TCP. In Section 5.4 on the limitations of TCP analysis we noted that there are many versions. Our analytical model with extra extensions can be used to model almost all TCP versions.
- The limitation of many TCP parameters can be overcome by our analytical model. In the model you can incorporate many TCP parameters as possible. The only limitation in this respect might be the protocol complexity.

In the results we have shown that at a lower congestion rate differentiated services is possible only if the wireless error rate is less than 1E-3. Else it's not possible to differentiate different service classes in a congested network. With the simulation and analytical results we have shown how elusive it is to provide absolute differentiated services. We have seen that sources with a lower reservation rate are greedier for the excess bandwidth than those with a relatively higher reservation rate.

CHAPTER 6
CONCLUSION

TCP and UDP are the main transport protocols used on the Internet today, up to now no better protocol has come up that surpasses the services offered by the TCP Protocol. In this thesis we focused on the transmission control protocol behavior with the following conditions; a lower IP layer incorporating QoS features and with a correlated loss wireless channel. The motivation of this work was the lack of any work combining the two aspects with respect to TCP behavior.

After looking at the need for analyzing the TCP protocol over wireless with DiffServ in Chapter 1 we looked at the QoS on the Internet in Chapter 2. We looked at the various mechanisms of achieving QoS in the Internet. QoS on the Internet can be realized by a combination of various mechanisms. We looked at the various buffer management, queue management and scheduling mechanisms that can be used to provide QoS on the Internet. The IETF has proposed the methods of providing QoS on the Internet, the best being differentiated services. We looked at the behavior of various proposed differentiated services classes on the Internet. We showed with simulation results that the assured

service class achieves a higher throughput than the best effort traffic. We also showed that on a combined channel with assured and premium traffic, premium traffic achieves least delay than the other traffic types. We also looked at the issue of relative and absolute differentiated services on the Internet and found that it is difficult to provide absolute differentiated services on the Internet.

In Chapter 3, we focused on the Transmission Control Protocol. First we developed our analytical model of the TCP Protocol. With our analytical model and other models we analyzed the behavior of the transmission control protocol with respect to its own parameters. We showed that TCP performance degrades as;

- the number of packet losses increases.
- round trip time is increased, a phenomena which is important in satellite channels.
- The timeout value is increased.
- As the maximum window size is reduced.

Furthermore the results were verified by simulation and we found out that our analytical model is a good model for the TCP protocol.

In Chapter 4 we looked at the performance of the TCP protocol over a wireless channel, we focused on the ways of improving TCP performance over wireless channels. One of the ways has been to come up with new protocols. We compared the performance of three TCP protocols, Tahoe, Reno and New Reno and found that TCP New Reno performs better that Reno which performs better that TCP Tahoe. We also showed how the window size could actually be a measure of the throughput of a TCP connection. We traced the window sizes of Tahoe, Reno and New Reno on a network at various packet losses. We showed that the window size of the New Reno traffic class was relatively higher than Reno which was relatively higher than Tahoe at a packet loss probability of less than 1E-3. The window sizes agreed with the throughput that the three classes received relative to each other. However, as the packets loss probability went beyond this we showed that the window sizes were almost the same and so was the degraded throughput. Other methods of improvement of TCP over wireless are the use of a Snoop protocol, and the ECN method. We adapted the ECN protocol to operate on a wireless

channel and compared its performance with the Snoop protocol and ordinary TCP Reno protocol. We showed that both the Snoop protocol and the ECN protocol showed a performance improvement over ordinary TCP Reno on a wireless channel when the packet loss probabilities are not so high. However, when both the protocols were compared we found out that Snoop TCP showed better performance than ECN TCP.

In Chapter 5, we presented our analysis of TCP over wireless with Differentiated Services. We presented our version of the packet train model of analyzing TCP over wireless with DiffServ. We modeled many traffic sources in a network with differentiated services. We came up with a variety of analytical and simulation results of the behavior of TCP Tahoe and Reno over a wireless channel with DiffServ. We then had both Tahoe and Reno sources on the network. In both the cases we drew the following conclusions.

- Assured differentiated services on wireless networks can be achieved when the packet loss probability is less than 1E-3.
- Losses with a smaller reservation rate consume a large proportion of excess bandwidth than the sources with a higher reservation rate.

6.1 Future Research Direction

Our investigation and results throw open several interesting issues for future research.

We have shown that both assured and premium DiffServ schemes are possible on the network. We have shown that two traffic classes can be given different QoS during network congestion relative to each other. This is what is called relative differentiated service. We need to investigate whether we achieve the required QoS during congestion and whether we can quantify the achieved QoS. We need further research in delivering absolute differentiated services.

During our analysis of the RED using the $M/D/1/K$ with state dependent arrivals we discovered a shortage in literature concerning this aspect. In queuing theory this aspect which seems to be obvious has not been exclusively looked at. In particular for the RED

algorithm the dropping of the IN or OUT packets may depend on the number of the IN or OUT packets in the queue. With the FCFS it means that you need to keep track of the number of packets of each type in the queue and of the order in which they occupy the buffer. This tends to make the model intractable and has been given little attention up to now. This area needs to be further researched into for it is handy in analyzing the RED algorithm which is the most favorable for introducing differentiated services in the networks.

We have also established techniques to compute the throughput when multiple TCP flows interact with a single bottleneck buffer. However the whole network scenario cannot be properly modeled by a single queue. We need to extend this work to consider a network of queues and derive the average dropping probabilities in this case.

Finally in our queues we did not specify the practical parameters that can be feasible on the networks. Specific research is to be undertaken to determine the RED queue parameters which can provide the optimum operating point for differentiated services. Further we need to look at the emerging queue management algorithms like BLUE, SRED and fully classify their operations and deploy them in the Internet.

Lastly we have shown that it is virtually impossible to provide differentiated services with TCP with a slight increase in congestion. There is need for incorporating one of the mechanisms of improvement of TCP for it to be possible to provide differentiated services. Alternatively fresh ideas in terms of improvement of TCP or a totally different protocol is required for DiffServ on the Internet.

APPENDIX 1. IP V.4 AND IPV6 HEADER SPECIFICATION

Version	IHL	Type of Service	Total Length	
Identification			Flags	Fragment Offset
Time To Live		Protocol	Header Checksum	
Source Address				
Destination Address				
Options				Padding

Figure A1. Ipv4 Header

Table A1. Ipv4 Header Fields Functions

Version	4 bits	Format of Internet header Ipv4/Ipv6
IHL	4 Bits	32 word bit length of header
Type of Service		Provides indication of QoS parameters
Total Length	16 bits	Total length of datagrams (octets)
Identification		Sender value that aids in assembling the fragments of a datagram
Flags	3 bits	Various Control Flags. Bit 0: reserved, Bit 1: Fragment or not, Bit 2: Whether last fragment.
Fragment Offset	13 bits	Indicates where in the datagram this fragment belongs.
Time to Live	8 bits	Indicates the maximum time the datagram is allowed to remain in the Internet system

Protocol	8 bits	Indicates the next level protocol used in the data portion of the Internet datagram.
Header checksum	16 bits	Checksum on the header only.
Source address	32 bits,	The source address.
Destination address	32 bits,	The destination address
Options	Variable	Include: End of option list, security, loose source routing, and strict source routing, record route stream id and internet timestamp.
Padding	Variable	Used to ensure that the internet header ends on a 32-bit boundary.

Version	Traffic Class	Flow Label	
Payload Length		Next Header	Hop Limit
Source Address			
Destination Address			

Figure A1.2. Ipv6 Header Specification

APPENDIX 2: TCP HEADER

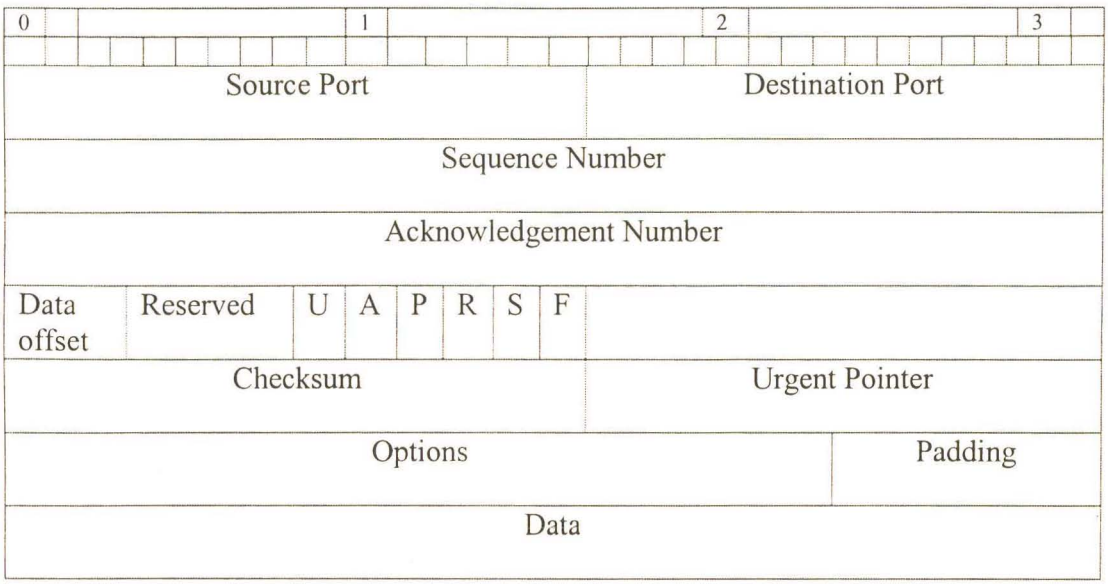


Figure A2.1. TCP Header

Table 2 TCP header specification

Source port:	16 bits	The source port number
Destination port:	16 bits	The destination port number
Sequence Number (SEQ):	32 bits	The sequence number of the first data octet in this segment (except when SYN is present) If SYN is present the sequence number is the initial sequence number (ISN) and the first data octet is ISN+1
Acknowledgement Number (ACQ):	32 bits	If the ACK control bit is set this field contains the value of the next sequence number the sender of the segment is expecting to receive. Once a connection is established this is always sent.

Data Offset:	4 bits	The number of 32 bit words in the TCP header. This indicates where the data begins. The TCP header (even one including options) is an integral number of 32 bits long.
Reserved:	6 bits	Reserved for future use. Must be zero.
Control bits:	6 bits (from left to right)	
U	URG:	Urgent Pointer field significant
A	ACK:	Acknowledgement
P	PSH:	Push function
R	RST:	Reset the connection
S	SYN:	Synchronize sequence numbers
F	FIN:	No more data from sender
Window:	16 bits	The number of data octets beginning with the one indicated in the acknowledgement field which the sender of this segment is willing to accept.
Checksum:	16 bits	Checksum field is calculated to verify the data correctness.

APPENDIX 3: STATE DEPENDENT ARRIVAL POISSON PROCESS

The RED queue is modeled as an $M/D/1/K$ queue with balking and is solved using the embedded Markov chain approach. For the single step transition matrix we need to determine the probability that k packets have arrived from state i to state j. The arrival rate is not constant. Let X_n be the n^{th} interarrival time, S_n be the arrival time of the n^{th} packet and λ_n be the arrival rate in the n^{th} interarrival time. From the RED algorithm, the interarrival times X_n are independent but are identically distributed up to r after which they are not identically distributed random variables. This is because RED starts dropping packets after the buffer is r (half full) else it accepts all packets. The probability distribution of k arrivals is the distribution of the interarrival times $X_1, X_k ....X_k$ and therefore we seek to characterize the sum of the interarrival times which is given by

$$S_k = f(x) + g(x) \qquad (A3.1)$$

where

$f(x) = X_1 + X_2 + X_r$ Independent and identically distributed

$g(x) = X_{r+1} + X_{r+2} ... + X_k$ Independent variables

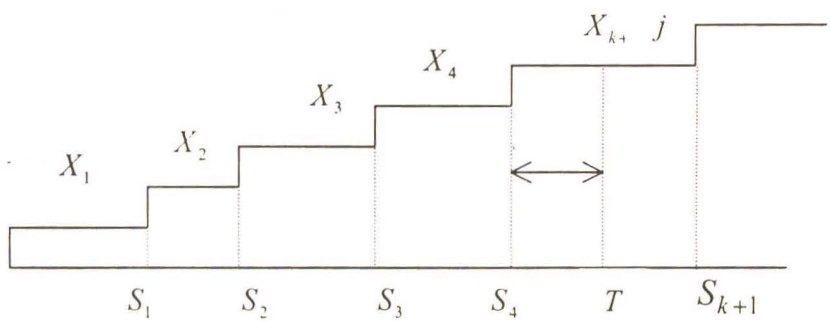


Figure A3.1. Interarrival times

To get the probability that k packets arrive within time t we use the arrival times. In our case the time is the fixed interarrival time and we denote it by T. This is depicted in figure A3.1. This probability is given by

$$P(S_k \le T, S_{k+1} > T) \qquad (A3.2)$$

$$= P(S_{k+1} > T) P(S_k \le T)$$

From the theory of the convolution of independent random variable we note that the moment generating function (MGF) of a sum of independent random variables is the product of the MGF of these random variables. We note that the Laplace transform plays a similar role for non-negative continuous random variables as the moment generating function does for discrete random variables. Therefore the laplace transform of equation (A3.1) is given by

$$L\left(S_k\right) = L(f(t))L(g(t)) \tag{A3.3}$$

$$L\left(S_k\right) = \left(\frac{\lambda}{\lambda + s}\right)^r \prod_{n=r+1}^{k} \frac{\lambda_n}{\lambda_n + s}$$

where the first part is when the arrivals are constant and the other part is for the variable arrivals. By splitting into partial fractions, the laplace transform is given by

$$\begin{aligned} L\left(S_k\right) = & \frac{\Phi_{A1}}{\lambda + s} + \frac{\Phi_{A2}}{(\lambda + s)^2} + \frac{\Phi_{A3}}{(\lambda + s)^3} + \ldots + \frac{\Phi_{Ar}}{(\lambda + s)^r} \\ & + \frac{\Phi_{B(r+1)}}{(\lambda_{r+1} + s)} + \frac{\Phi_{B(r+2)}}{(\lambda_{r+2} + s)} + \ldots + \frac{\Phi_{B(k-1)}}{(\lambda_{k-1} + s)} + \frac{\Phi_{Bk}}{(\lambda_k + s)} \end{aligned} \tag{A3.4}$$

Taking the inverse Laplace transform of (A3.4) we get the distribution as below

$$P_{S_k}(t) = \sum_{i=1}^{r} \frac{\Phi_{Ai} t^{i-1} e^{-\lambda t}}{(i-1)!} + \sum_{i=r+1}^{k} \Phi_{Bi} e^{-\lambda_i t} \tag{A3.5a}$$

where

$$\Phi_{Bi} = (\lambda_i + s) L\left(S_k\right)\Big|_{s=\lambda_i} \tag{A3.5b}$$

$$\Phi_{Ai} = \frac{d^{r-i}}{ds^{r-i}} \left[(\lambda + s)^r L\left(S_k\right)\right]\Big|_{s=\lambda_i} \tag{A3.5c}$$

Challenge 1: There is a problem of finding the derivatives in equation (A3.5c) in the program as the queue size increases. The solution to this problem was to change the RED algorithm so that the arrivals will be continually varying from the beginning though by different margins for the IN and OUT packets. Thus the RED was modified as shown in figure A3.2. Following the modification of the RED algorithm Equations (A3.5a) reduces to the equation of probability density function given by

$$P_{S_k}(t) = \sum_{i=1}^{k} \Phi_{Bi} e^{-\lambda_i t} \qquad (A3.6a)$$

where

$$\Phi_{Bi} = (\lambda_i + s) L(S_k)\Big|_{s=-\lambda_i} \qquad (A3.6b)$$

and our probabilities as in equation A3.2 can be given by

$$P(S_k \le T) = -\sum_{i=1}^{k} \frac{\Phi_{Bi} e^{-\lambda_i t}}{\lambda_i} \Bigg|_0^T \qquad (A3.7)$$

The above distribution of a convolution of independent random exponential variables with different rate gave a hypoexponential distribution.

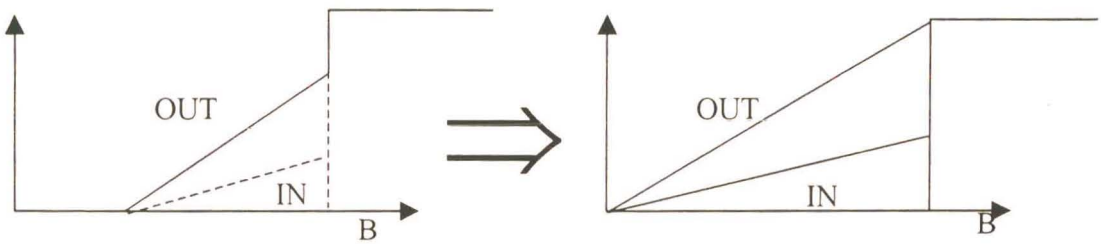


Figure A3.2. Modification of the RED Algorithm

Challenge 2: The pdf of the hypoexponential distribution (A3.6) is not trivial to compute. It was plotted and observed as the number of phases increased it gives negative values. The same problem has been encountered in the solution of the convolution of exponential random variables in [62]. It is noted that the coefficient (A3.6b) are not probabilities since some of them are negative. The other way of looking at this pdf is as the time to absorption of an $M/D/1/K$ queue [63] but still it is not trivial to compute the distribution. Therefore faced with these two challenges we chose to model our arrival rate as follows. Let λ_k be the interarrival rate within the interarrival time X_n. We use the average arrival rate as given by

$$\lambda_k^{av} = \frac{\sum_{l=t}^{l=k} \lambda_l}{k} \qquad (A3.11)$$

where λ_l is the overall arrival rate in the queue before balking.

APPENDIX 4:SIMULATION ALGORITHM

FLOW CHART

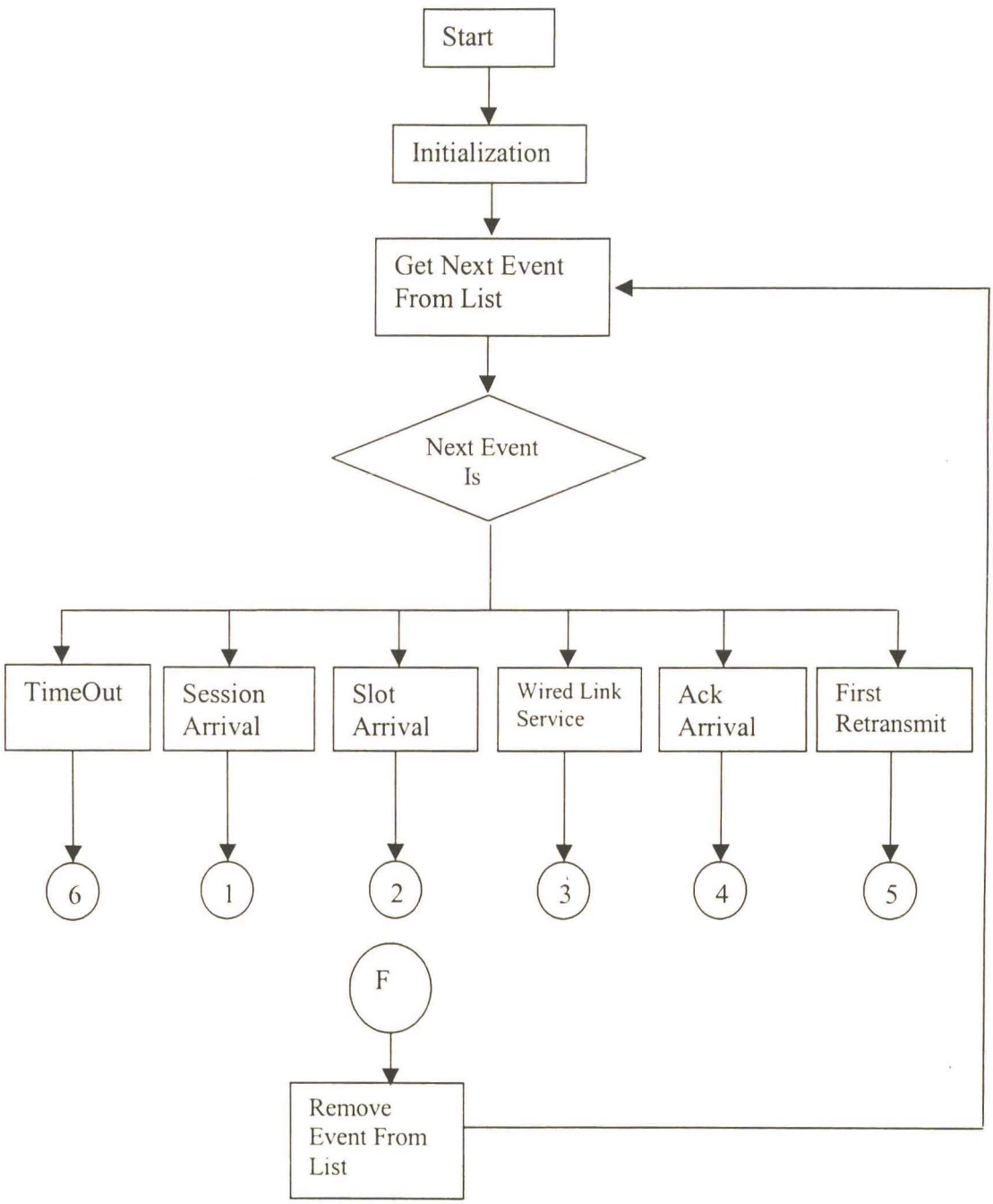


Figure A4.1. Simulation Algorithm

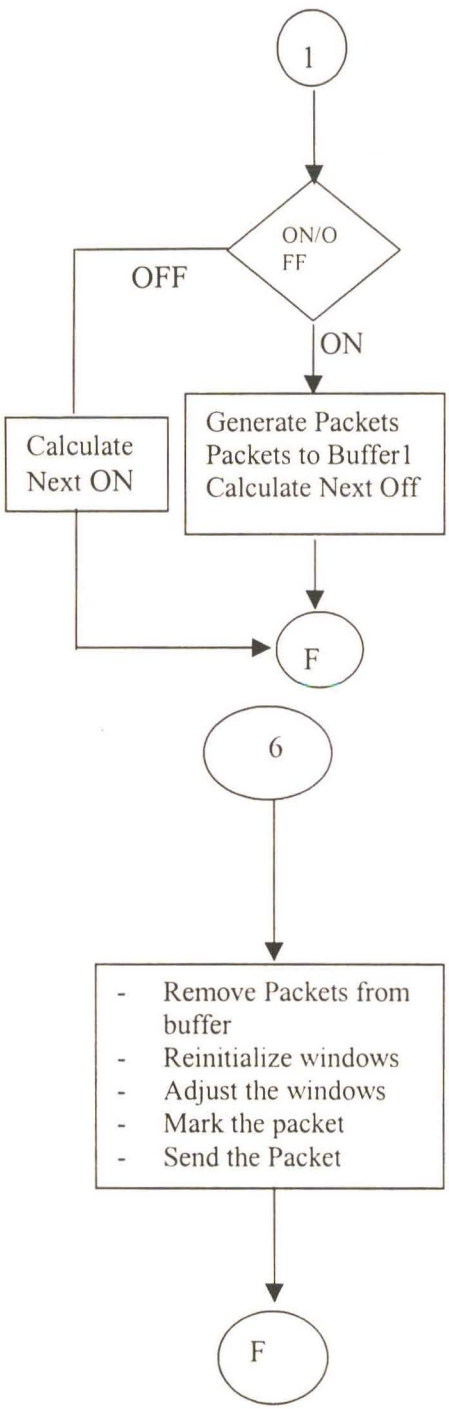


Figure A4.2. Session Arrival

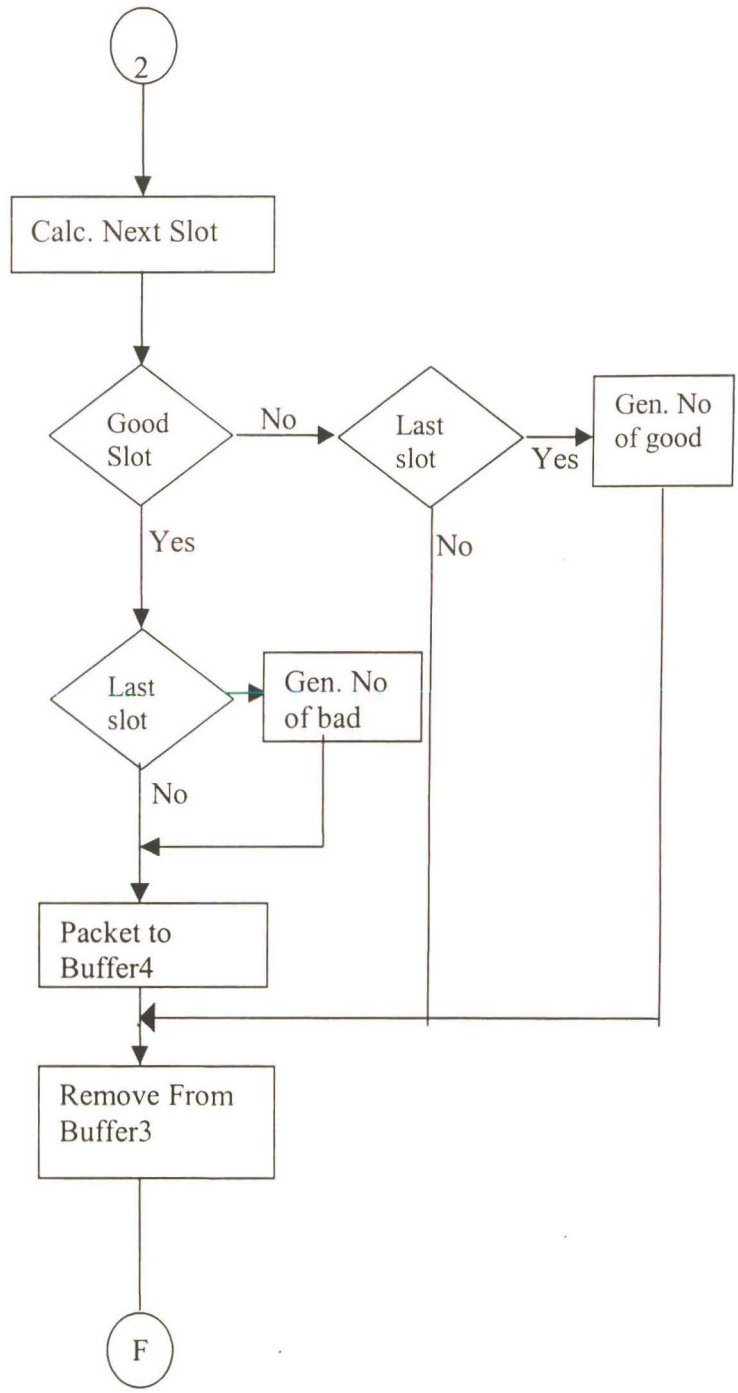


Figure A4.3. Slot Arrival

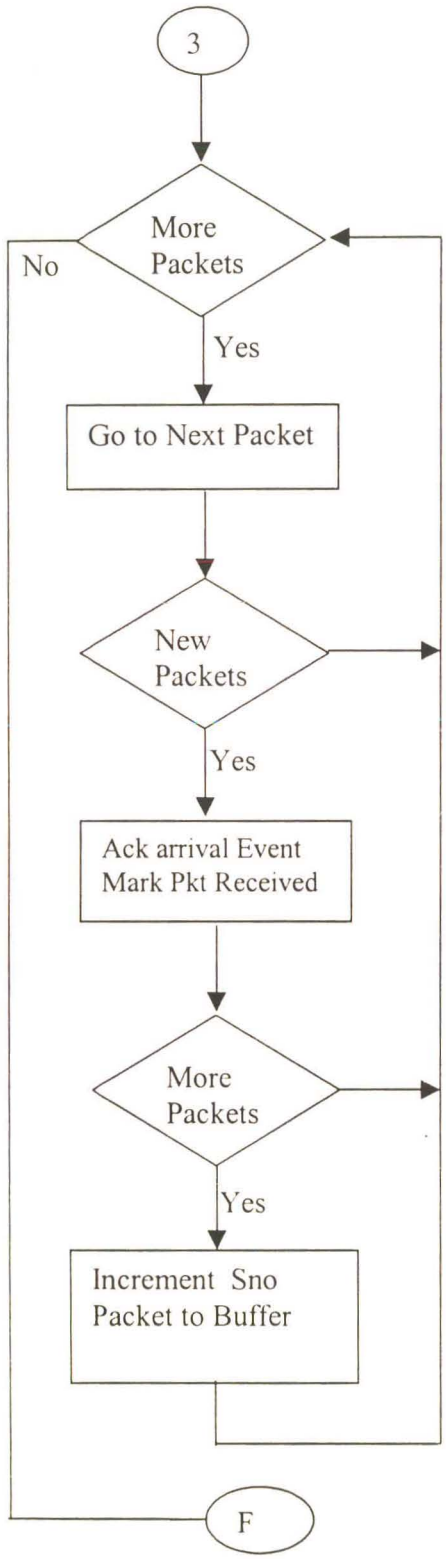


Figure A4.5. Wired Link Service

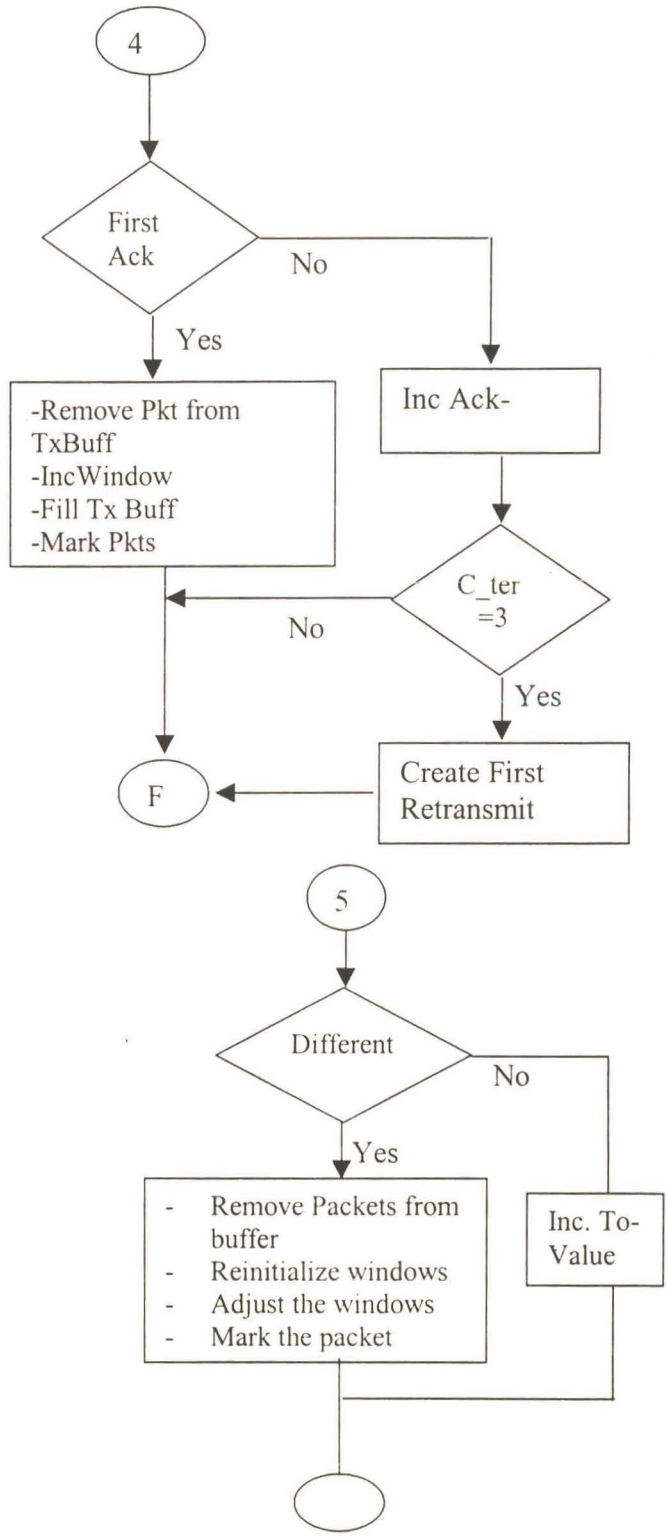


Figure A4.7. Ack Arrival

REFERENCES

[1] K. Thompson, G. J. Miller, and R. Wilder, "Wide-Area Internet Traffic Patterns and characteristics," *IEEE Network,* November, 1997

[2] Clark, D., "Adding Service Discrimination to the Internet," *LCS MIT Technical Report*, September 1995.

[3] K. Nichols, V. Jacobson, L. Zhang, "A Two-bit Differentiated Services Architecture for the Internet," *RFC 2638,* July 1999.

[4] Nagle J., "On Packet Switches with Infinite Storage," *IEEE Transactions on Communications,* vol. 35, 1987, pp. 435-38.

[5] Branden B, et al., "Recommendations on Queue Management and Congestion Avoidance in the Internet," *RFC 2309*, IETF, April 1998.

[6] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Trans on Networking,* Vol. 1, pp. 397- 413, August 1993.

[7] W.C. Feng et al., "A self-Configuring RED Gateway," *Proceedings IEEE INFOCOM*, New York, March 1999.

[8] T.J. Ott, T.V.Lakshman, and L.H. Wong, "SRED: Stabilized RED," *Proceedings of IEEE INFOCOM,* pp. 1346-1355, March 1999.

[9] W.C. Feng et al., "BLUE: A New class of active queue management algorithms," *Proceedings IEEE INFOCOM,* New York, March 1999.

[10] M. Markaki, E. Nikolouzou, I. Venieris, "Performance Evaluation of Scheduling Algorithms for the Internet," *IFIP 8th Workshop Proceedings*, July 2000.

[11] D. Clark, S. Shenker, L. Zhang, "Supporting Real-Time Applications in an Integrated Services Packet Network: Architecture and Mechanism," *Proceedings of ACM SIGCOMM'92*, pp 14-26, August 1992.

[12] A. Parekh, R. Gallager, "A Generalised Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single-Node Case," IEEE/ACM *Trans. on Networking*, vol. 1, June 1993.

[13] D. Stephens, J. Bennett, H. Zhang, "Implementing Scheduling Algorithms in High-Speed Networks," *IEEE selected Areas in Communications*, Vol. 17, no.6, June 1999.

[14] S. Shenker, R. Branden, D. Clark, "Integrated Services in the Internet Architecture: an overview," *RFC 1633*, June 1994

[15] Tsipora P et al, "Design and Implementation of an RSVP-Based Quality of Service Architecture for an Integrated Services Internet," *IEEE selected Areas in Communications*, Vol. 16, no.3, April 1998.

[16] Branden, R., et al., "RSVP: A New Resource ReSerVation Protocol Protocol," *IEEE Network*, September 1995.

[17] S. Blake et al., "An Architecture for Differentiated Services," *IETF RFC 2475,* Dec 1998

[18] Zheng Wang, "User-Share Differentiation (USD) Scalable bandwidth allocation for Differentiated Services, *Internet draft,* November 1997.

[19] J. Heinanen, F. Baker, W. Weiss and J. Wroclawski, "Assured Forwarding PHB Group," *Internet RFC*, June 1999, RFC 2597

[20] David Clark and Wenjia Fang, "Explicit Allocation of Best-Effort Packet Delivery Service", *IEEE/ACM Trans. Networking*, vol.6, no.4, pp.362-373, Aug. 1998.

[21] Tom Walingo and Fambirai Takawira, "Differentiated Services over GPRS," SATNAC 2000 conference, CapeTown, South Africa.

[22] C. Dovrolis and P. Ramanathan, "A Case for Relative Differentiated Services and the Proportional Differentiation Model," *IEEE Network,* September/October 1999.

[23] Saleem Bhatti and Jon Crowcroft, "QoS-Sensitive Flows," *IEEE Internet Computing,* July-August 2000.

[24] A. Chockalingham, "Performance of TCP/RLP Protocol Stack on Correlated Fading DS-CDMA Wireless Links," *IEEE/VTC.* Vol.49, No1, Jan 2000.

[25] Balakrishnan H., et al, "Comparison of Mechanisms of Improving TCP Performance over Wireless Links," IEEE/ACM SIGCOMM'96, August 1996.

[26] T.V. Lakshman and U. Madhow, "The performance of TCP/IP for networks with high bandwidth delay products and random loss," ", *IEEE/ACM Trans. Networking*, vol. 5, no.3, pp.336-360, June 1997.

[27] M. Mathis, J. Semke, J. Mahdavi and T. Ott, "The Macropscopic behaviour of the TCP congestion avoidance algorithm," *Computer Communication Review*, vol. 27, no. 3, July 1997.

[28] J. Padhye, V.Firoiu, D. Towsley and J. Kurose, "Modelling TCP Reno Performance: A simple Model and its Empirical Validation", *IEEE/ACM Trans. Networking*, vol.8, no.2, pp.133-145, April 2000.

[29] W. R. Stevens, "TCP/IP Illustrated," vol. 1. Reading, MA: Addison Wesley, 1994

[30] Anurag Kumar, "Comparative Performance Analysis of Versions of TCP in Local Network with a Lossy Link," *IEEE/ACM Trans. Networking*, vol.6, no. 4, pp.485-498, Aug.1998.

[31] Richard J et al, "Analysis and Comparison of TCP Reno and Vegas," *IEEE Infocom'99, March 1999.*

[32] S. Floyd and K. Fall, "Promoting the Use of End-to-end Congestion Control in the Internet,' *IEEE/ACM Transaction on Networking,* vol. 7, no. 4, August 1999, pp. 458-72.

[33] Farooq M. Anjum and Leandros Tassiulas, "An Analytical Model for the Various TCP Algorithms Operating Over a Wireless Channel," *IEEE WCNC,* 1999.

[34] I.Yeom and A.L.N. Reddy, "Modelling TCP Behaviour in a Differentiated-Services Network", *IEEE/ACM Trans. Networking*, vol. 9, no.1, pp.31-46, February 2001.

[35] Alhussein A. Abouzeid, Sumit Roy and Murat Azizoglu, "Stochastic Modeling of TCP over Lossy Links," *Proceedings of INFOCOM'00,* March 2000

[36] M. Zorzi and R. Rao, "Effect of correlated errors on TCP" Proc. CISS, pp. 666-671, March 1997.

[37] M. Zorzi, A. Chockalingam, A. Rao, "Throughput Analysis of TCP on channels with Memory", *IEEE Selected areas in Communications*, vol.18, no.7, pp.1289-1300, July 2000

[38] Chaskar et al, "TCP Over Wireless with Link Level Error Control: Analysis and Design Methodology", *IEEE/ACM Trans. Networking*, vol.7, no.5, pp.605-614, Oct. 1999.

[39] J. D. Parsons, "The Mobile Radio Propagation Channel," Pentech Press, London, 1992,

[40] Hong Shen Wang and Nader Moayeri, "Finite-State Markov Channel—A Useful Model for Radio Communication Channels," *IEEE Trans on Vehicular Technology*, Vol. 44, No. 1 pp.163-172, February 1995.

[41] Fulvio Babich and Giancarlo Lombardi, "A Markov Model for the Mobile Propagation Channel," *IEEE Tran. on Vehicular Technology*, Vol. 49, No. 1 pp.63-73, January 2000.

[42] Michael Zorzi, Ramesh Rao and Laurence Milstein, "On the Accuracy of a first-order Markov model for data transmission on fading channels," *ICUPC'95*, Tokyo, Japan, November 1995.

[43] Christopher Tan and Norman Beaulieu, "On First-Order Markov Modeling for the Rayleigh Fading Channel," IE*EE Trans. on Communications,* Vol. 48, No. 12 pp.2032-2040, December 2000.

[44] Leif Wilhelmsson and Laurence Milstein, "On the Effect of Imperfect Interleaving for the Gilbert-Elliott Channel," *IEEE Trans. on Communications,* Vol. 47, No. 5, pp.681-688, May 1999.

[45] Chockalingam and Gang Bao, "Performance of TCP/RLP Protocol Stack on Correlated Fading DS-CDMA Wireless Links", *IEEE Trans VTC,* Jan 2000, Vol. 49, no.1, pp. 23-33

[46] Conolly T. et al, "An Extension to TCP: Partial Order Service," RFC 1693, 1994.

[47] Balakrishnan H. et al, "Improving TCP/IP Performance over Wireless Networks," *IEEE/ACM Mobicom*, 1995.

[48] A. Bakre and B. R. Badrinath, "Implementation and Performance Evaluation of Indirect-TCP," *IEEE Trans. Comp.* , Vol.46, no.3, Mar.1997, pp.260-78.

[49] Allman, M., Floyd, S. and C. Partridge," Increasing TCP's Initial Window", *RFC 2414*, September 1998.

[50] Allman, M et al, "Enhancing TCP Over Satellite Channels using Standard Mechanisms" *RFC 2488.*

[51] Jia Ma et al, "An enhanced TCP mechanism-Fast-TCP in IP networks with wireless links" *Baltzer AG, Wireless Networks* 6(2000), pp. 375-379.

[52] Allman, M et al, " Ongoing TCP Research Related to Satellites", RFC 2760.

[53] S. Wang and H. Kung, "Use of TCP Decoupling in Improving TCP Performance over Wireless Networks" *Baltzer AG, Wireless Networks 7* (2000), pp. 221-236.

[54] K. K. Ramakrishan et al, "A Proposal to add Explicit Congestion Notification (ECN) to IP," *RFC 2481*, Jan 1999.

[55] Martin May et al, "Simple Performance Models of Differentiated Services Schemes for the Internet", *IEEE Infocom '99*, March 1999, pp. 1385-94.

[56] I. Yeom and A. L. Narasimha Reddy, "Realizing throughput guarantees in a differentiated services network," *Proceedings of ICMCS*, pp. 372--376, vol. 2, June 7-11, 1999.

[57] D. Gross and C. Harris, "Fundamentals of queuing Theory," John Wiley and Sons, Inc., Canada and USA, 1974.

[58] R. B. Cooper, "Introduction to queuing theory," McMillan Company, New York, 1972.

[59] Claudio casseti and meao, "A new approach to Model the Stationary Behavior of TCP Connections," Pro. IEEE Infocom March 2000. pp. 367-375.

[60] Seddigh Nabil et.al, "Bandwidth Assurance Issues for TCP flows in a Differentiated Services Network", *IEEE Globecomm '99*, pp. 1792-1798.

[61] Abdelnaser Adas, "Traffic Models in Broadband Networks," *IEEE Communications Magazine*, July 1997.

[62] *S.M. Ross,* "Introduction to Probability Models," *sixth edition, Academic Press,* San Diego, CA 1997.

[63] Kishor S. Trivedi, "Probability and Statistics with Reliability, Queuing, and Computer Science Applications," Prentice-Hall, INC., Englewwood Cliffs, New Jersey 07632.

[64] European Telecommunications Standard Institute, ETSI HIPERLAN/1 standard, http://www.etsi.org/technicalactiv/hiperlan1. htm, Apr.2001.

[65] IEEE P802.11, The working Group for Wireless Local Area Networks, http://www.grouper.ieee.org/groups/802/11/, Apr.2000.

[66] J. Postel, Internet Protocol - DARPA Internet Program Protocol Specification, RFC 791, 1981

[67] Deering S and Hinden R., "Internet Protocol, Version 6, Specification," RFC1883, Xerox PARC, Ipsilon Networks Inc., December 1985.

[68] Sahu et.al, "A Quantitative Study of Differentiated Services for the Internet', *IEEE Globecomm '99*, pp. 1808-1817.

[69] Tom Walingo and Fambirai Takawira, "Comparison of Snoop and ECN Protocols," SATNAC 2001 conference, Wild Coast Sun, South Africa.

[70] C. Perkins, Mobile IP Design and Practices, Addison-Wesley, 1998.

[71] A. Campbell and J. Gomez, "An Overview of Cellular IP," *IEEE WCNC 1999*, vol. 2, pp. 606-610.

[72] M. Mathis and J. Mahdavi, "Forward Acknowledgement: Refining TCP Congestion Control," *Proc. ACM SIGCOMM*, Aug. 1996.

[73] C. Parsa and J. Garcia-Luna-Aceves, "Improving TCP Congestion Control over Internets with Heterogeneous Transmission Media," *Proc. IEEE ICNP'99* Toronto, Oct. 1999