The Modelling of TCP Traffic in MPLS Networks

Marcel Villet

Thesis presented in partial fulfilment
of the requirements for the degree of
Master of Science
at the University of Stellenbosch

Supervisor: Prof. A. E. Krzesinski

April 2003

Declaration

I, the undersigned, hereby declare that the work contained in this thesis is my own original work and has not previously in its entirety or in part been submitted at any university for a degree.

Signature: Date:

Abstract

The Internet has experienced tremendous growth in the last three decades and has emerged as a platform to carry all forms of communications including voice, video and data. Along with this growth came the urgency for quality of service (QoS) controls in IP networks as different types of traffics have different service requirements. Although the IP protocol is able to scale to very large networks, it does not provide sufficient functionality for traffic engineering in order to enable QoS control.

Multi-protocol label switching (MPLS) is a new routing technology that enhances IP with some QoS concepts from ATM and uses relatively simple packet forwarding mechanisms. MPLS has the ability to perform traffic engineering and QoS control by routing traffic flows on virtual connections called label switched paths (LSPs) which are assigned capacity.

A large portion of the traffic carried on the Internet consists of data traffic in the form of TCP traffic. This thesis investigates several TCP models to find the ones most suitable to represent TCP traffic in MPLS networks. The models consist of three types. The first type models a single TCP source and the second type models a fixed number of TCP sources. The third type models an infinite number of TCP sources. The models were evaluated by comparing their throughput predictions and results obtained from simulation experiments that were done with the widely-used simulator *ns*. We also present a simple derivation of the $1/\sqrt{e}$ law for the TCP congestion window size where e is the packet loss probability.

Opsomming

In die afgelope drie dekades het die Internet beduidende groei ervaar, soveel so dat dit ontluik het as 'n medium om alle tipes van moderne kommunikasies te hanteer insluitend telefoon, video en data. Hierdie groei het gepaard gegaan met die behoefte na diensvlak (QoS) meganismes in IP netwerke aangesien verskillende tipe kommunikasies verskillende diens vereistes het. Alhoewel die IP protokol skalleerbaar is tot baie groot netwerke, voorsien dit nie voldoende funksionaliteit om QoS beheer toe te pas nie.

"Multi-protocol label switching" (MPLS) is 'n nuwe roeterings tegnologie wat IP aanvul met QoS konsepte van ATM en dit maak gebruik van relatief eenvoudige pakkie versendings-meganismes. MPLS het die vermoë om netwerk-verkeer reëling en QoS beheer toe te pas deur verkeers-strome te roeteer op virtuele roetes genaamd "label switched paths" (LSPs) aan wie kapasiteit toegeken is.

'n Beduidende gedeelte van Internet-verkeer bestaan uit TCP-verkeer. Hierdie tesis ondersoek verskillende modelle van TCP om dié te vind wat die mees geskik is om TCP verkeer in MPLS netwerke te verteenwoordig. Drie tipes modelle is ondersoek. Die eerste tipe moduleer 'n enkele TCP verkeersbron en die tweede tipe moduleer 'n vasgestelde aantal TCP verkeersbronne. Die derde tipe moduleer 'n oneindige aantal verkeersbronne. Die modelle is geëvalueer deur hul voorspellings van die tempo van data transmissie te vergelyk met resultate van simulasies. Die simulasies is gedoen met die veelgebruikte simulator *ns*. Hierdie tesis bevat ook 'n eenvoudige afleiding vir die $1/\sqrt{e}$ wet vir die TCP oorlading venster grootte met e die verlies waarskeinlikheid van 'n netwerk pakkie.

Acknowledgements

This work was performed within the *Siemens-Telkom Centre of Excellence for ATM & Broadband Networks and their Applications* and is supported by grants from the South African National Research Foundation, Telkom SA Limited and Siemens Telecommunications.

I also spent a sabbatical at the *Teletraffic Research Centre* at the University of Adelaide. My host was Prof PG Taylor from the Department of Applied Mathematics at the University of Adelaide.

List of Publications

1. M Villet and AE Krzesinski. On the Accuracy of Two TCP Performance Models of MPLS Networks. In Proceedings of the *South African Telecommunications, Networks and Applications Conference (SATNAC) '02*, KwaZulu-Natal, South Africa, September 2002.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In the last two decades the Internet has experienced tremendous growth and it now carries all forms of modern communications including voice, video and data. This expansion has greatly increased the need for quality of service (QoS) controls in IP networks as different types of traffic have different service requirements. Real-time traffic such as voice over IP (VoIP) and video on demand requires transmission with low delay, and video requires much more bandwidth than VoIP. For non-real-time traffic such as data transfers, transmission must proceed with low loss rates but not necessarily with low delay. Although the IP protocol is able to scale to very large networks, it does not provide sufficient functionality for traffic engineering in order to enable QoS control. Packets are routed with the OSPF (open shortest path first) routing protocol and the only service class in an IP network is best-effort.

Multiprotocol label switching (MPLS) is a new routing technology that enhances IP with some QoS concepts from ATM and uses relatively simple packet forwarding mechanisms. MPLS has the ability to perform traffic engineering and QoS control by routing traffic flows on virtual connections called label switched paths (LSPs) which are assigned capacity. Traffics with different service requirements can thus be treated accordingly. Real-time traffic may be routed along shorter paths with lower delay while non-real-time traffic may be routed along longer paths with higher delay and lower loss rates. MPLS is discussed in more detail in section 1.1.

A large portion of the traffic carried on the Internet consists of data traffic in the form of TCP traffic as TCP is the protocol used by many applications that handle file and web transfers. Studies of TCP's performance were originally based on simulation experiments and TCP traffic trace measurements. In recent years, several analytic models of TCP's performance were developed to gain insight into the characteristics of TCP behaviour in many environments. A large number of TCP models are documented in the literature and many offer different insights into its behaviour.

The purpose of this thesis is to investigate several models of TCP in order to find those that are most suitable to describe TCP traffic in MPLS networks. Specifically we are interested in models of implementations of TCP, such as TCP Tahoe and TCP Reno, which are derived from the Berkeley Software Distribution (BSD) releases. Twelve models were chosen for closer scrutiny of which two were extended to better describe the burstiness of TCP traffic. The models are of three types. The first type models a single TCP source and the second type models a fixed number of TCP sources. The third type places no constraint on the number of TCP sources and assumes an infinite number of TCP sources. The models were evaluated by comparing their throughput predictions versus the results obtained from simulation experiments that were done using the widely-used simulator *ns*. The results in this thesis indicate which models are best suited to predict certain network performance metrics for the various network scenarios that were investigated. The work is inspired by [74] which compares the throughput predictions of three TCP models [30, 62, 73] and simulation results for the scenario where N ON/OFF TCP sources share bandwidth on a router.

This thesis is organised as follows. Chapter 2 provides a concise description of TCP and its relevant algorithms, and we briefly examine the various versions of TCP. The necessary mathematical background is given in chapter 3 that is used by the TCP models. The TCP models are discussed in chapters 4 to 6 and we only give an overview of the way in which the TCP mechanisms are included in the models. The necessary equations for calculating the packet throughput per TCP connection are also given. Section 6.2 presents a simple derivation of the $1/\sqrt{e}$ law for the TCP congestion window size where e is the packet loss probability. In chapter 7 we give details regarding the simulation experiments, followed by the performance results. The conclusion follows in chapter 8. Additional mathematical background is presented in appendix A and appendix B contains a list of TCP models that were not considered for the MPLS network scenario. Additional information regarding simulation experiments is given in appendix C and the waiting time in a finite single server queue is derived in appendix D.

1.1 Multiprotocol Label Switching

Multiprotocol label switching (MPLS) (see Rosen *et al.* [69]) is the compound name for the corresponding IETF (Internet Engineering Task Force[1]) working group and their efforts regarding the MPLS protocol. MPLS makes use of a technology called label switching which has been implemented in one form or another by vendors such as Cisco, Ipsilon, Toshiba and IBM (see Davie *et al.* [17]). Label switching is implemented in routers to determine the next hop to a packet's destination. MPLS is *multiprotocol* as it can be implemented on many network hardware technologies such as ATM (Asynchronous Transfer Mode). The following is a simplified description of MPLS and serves only as background to the technology.

An MPLS backbone network consists of MPLS label switching routers (LSRs) which are connected by physical links. An LSR is called an ingress or an egress router depending on whether it is handling traffic that respectively enters or leaves the MPLS-capable part of the network. Traffic

[1]http://www.ietf.org

offered between an ingress and an egress LSR is carried on one or more label switched paths (LSPs). The backbone is further connected to other network domains via edge label switching routers (ELRs). This is illustrated in figure 1.1.

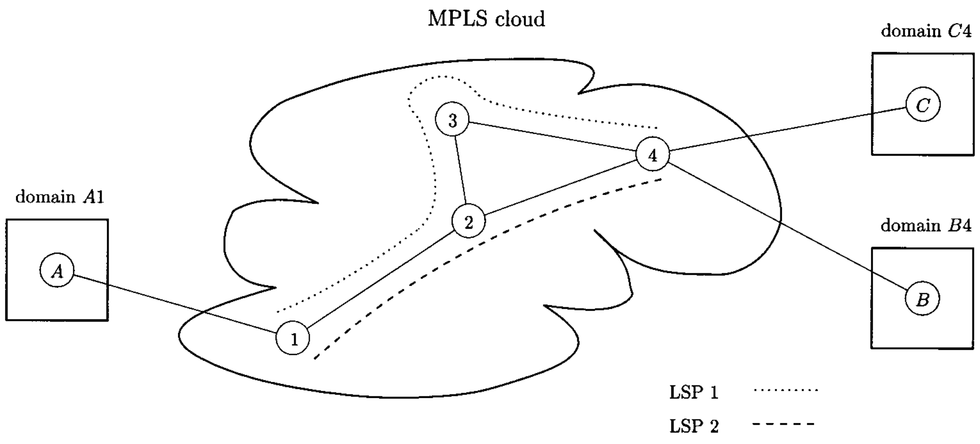


Figure 1.1: An MPLS backbone network connected to ELRs in outside domains.

In figure 1.1 the cloud represents the MPLS backbone network. Nodes 1 to 4 are LSRs which are connected as shown in the figure. The backbone is connected to domains $A1$, $B4$ and $C4$ via ELRs A, B and C respectively. Two LSPs, LSP 1 and LSP 2, carry traffic from LSR 1 to LSR 4. LSP 1 consists of links 1-2, 2-3 and 3-4 and LSP 2 consists of links 1-2 and 2-4.

MPLS categorises every packet into a forward equivalence class (FEC). A FEC is a group of packets with common attributes such as being transmitted between the same origin and destination (OD) pair and with the same forwarding treatment. In figure 1.1, TCP packets from FTP applications that travel from domains $A1$ to $C4$ may be in the same FEC (denote FEC $A1$-$C4$), and the same for TCP packets from FTP applications traveling between domains $A1$ and $B4$ (denote FEC $A1$-$B4$). FECs can be grouped into a single traffic trunk (flow) which is transmitted on an LSP through the backbone network. For example, FECs $A1$-$C4$ and $A1$-$B4$ can be grouped into a single traffic trunk and be transmitted on either LSP 1 or LSP 2. Alternatively, FECs $A1$-$C4$ and $A1$-$B4$ can be assigned to separate traffic trunks that are transmitted on LSP 1 and LSP 2 respectively or on LSP 2 and LSP 1 respectively.

LSPs are assigned virtual capacity and MPLS can perform LSP overload protection by means of connection admission control and packet policing at the ELRs. MPLS has mechanisms for managing LSPs, for example adding or removing LSPs and assigning capacity to LSPs as required. MPLS can also provide service separation for FECs with different service requirements such as low delay for real-time applications and low packet loss for data transfer applications.

The MPLS protocol thus provides the possibility to perform traffic engineering which is concerned with the performance optimization of the network. In order to design a set of LSPs such that the overall network performance is optimal, one needs to model the traffics in the network. The service separation feature of MPLS makes it possible to model each LSP separately with queuing taking place at the edge of the network at the ELRs. This thesis is concerned with finding suitable models to represent TCP traffic carried on an LSP. The network setup in section 6.1 represents an MPLS network setup consisting of two ELRs connected by two LSPs.

Although we applied the TCP models in this thesis to MPLS networks, they are also applicable to other network technologies, such as ATM networks, where a path between an OD pair in the network can be modelled as a single (logical) link. The insights offered by the comparisons between the *ns* simulation results and the predictions from the TCP models therefore extends further than MPLS networks.

1.2 Abbreviations

The following abbreviations are used throughout this thesis.

ACK	acknowledgment	Sect. 2.2
AIMD	additive increase and multiplicative decrease	Sect. 2.6.2
BSD	Berkeley Software Distribution	Sect. 2.8
CA	congestion avoidance	Sect. 2.6.1
DR	TD loss retransmission	Sect. 6.3
ELR	MPLS edge label switching router	Sect. 1.1
FACK	forward acknowledgment	Sect. 2.8.5
FT	fast retransmit	Sect. 2.6.2
FR	fast recovery	Sect. 2.6.2
FTP	File Transfer Protocol	Sects. 2.1 & 2.6
IP	Internet Protocol	Sect. 2.1
LSP	label switched path	Sect. 2.1
M	Markov	
M	Mega (1 million)	(when used with a unit of measurement)
MMPP	Markov modulated Poisson process	Sect. 3.5
MPLS	multi-protocol label switching	Sect. 1.1
MSS	maximum segment size	Sect. 2.2
PH	phase type	Sect. 3.3
PT	power-tail	Sect. 3.4
QBD	quasi birth-and-death	Sect. 3.10
RTT	round trip time	Sect. 2.4
s	seconds	(when used as a unit of measurement)
SACK	selective acknowledgment	Sect. 2.8.4
SM	semi-Markov	Sect. 3.5
SMTP	Simple Mail Transfer Protocol	Sects. 2.1 & 2.6
SS	slow start	Sect. 2.6.1
TCP	Transmission Control Protocol	Chap. 2
TD	triple duplicate acknowledgment	Sect. 2.6.2
TO	timeout	Sect. 2.6.2
TPT	truncated power-tail	Sect. 3.4
TR	timeout retransmission	Sect. 6.3

Chapter 2

An Overview of TCP

This chapter presents an overview of the Transmission Control Protocol (TCP). After a brief introduction to TCP/IP we examine the mechanisms of TCP which are relevant to the TCP models presented in this thesis. Only a brief overview of these mechanisms is given, and it should be noted that the TCP protocol is significantly more complex than is portrayed in this chapter. This chapter therefore serves only as background to the TCP models. The discussion follows the specification given in Stevens [75, 76, 77]. It is followed by short descriptions of various TCP versions and how these versions implement and extend the TCP mechanisms.

2.1 What is TCP, IP and TCP/IP?

The TCP/IP protocol suite allows computers from different vendors with different architectures and running different operating systems to communicate with each other. The protocol suite was initially developed in the late 1960's as a USA government funded research project into packet switching networks, and forms the basis for modern communications on the Internet.

TCP/IP consists of different protocols which are grouped in four layers as illustrated in Figure 2.1.

Application	Telnet, Rlogin, FTP, SMTP (e-mail), *etc.*
Transport	TCP and UDP
Network	IP
Link	device driver for network card

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

The Internet Protocol (IP) is implemented at the network layer and enables TCP segments[1] contained in IP packets to be sent across the Internet. TCP is implemented at the transport layer and provides a connection oriented, reliable end-to-end byte stream service. The term *connection oriented* means that for two applications to exchange data by means of the TCP protocol, a TCP connection must first be set up between the two applications. TCP provides reliability by means of features such as the controlling of the flow of data into the network and by keeping checksums and timers. TCP is used by many popular applications such as Telnet, Rlogin, FTP and electronic mail (SMTP) to transmit data.

2.2 Connection Establishment

A TCP connection is established between a sender and a receiver by means of a three way handshake. The sender sends a synchronize (SYN) packet to the receiver which in turn responds with a SYN packet that acknowledges the first SYN packet from the sender. The sender then acknowledges the receiver's SYN packet with an acknowledgment (ACK) packet.

During connection establishment, each end of the connection advertises a maximum segment size (MSS) which is the maximum number of bytes allowed in the data payload of the TCP packet. The MSS defaults to x if either the two ends of the connection are not on the same local Ethernet, or if one end does not receive an MSS indication. For many BSD implementations, the MSS must be a multiple of 512 bytes, and the default MSS advertised is 1024 bytes and $x = 512$ bytes. For other systems such SunOS 4.1.3, Solaris 2.2 and AIX 3.2.2, the default MSS advertised is 1460 bytes and $x = 536$ bytes.

A limit is placed on the MSS by the value of the maximum transmission unit (MTU) that is determined by the link layer protocol. A transmission unit (TU) excludes the header of the link layer protocol data unit and therefore consists of the IP header (20 bytes), the TCP header (20 bytes) and the TCP data. Table 2.1 lists MTU values for various link layer protocols.

Link Layer Protocol	MTU (bytes)
Hyperchannel	65535
16 Mbits/sec token ring (IBM)	17914
4 Mbits/sec token ring (IEEE 802.5)	4464
FDDI	4352
Ethernet	1500
IEEE 802.2/802.3	1492
X.25	576
Point-to-point (*e.g.* SLIP and PPP)	296

Table 2.1: Values of the maximum transmission unit (MTU) for various link layer protocols.

For example, for Ethernet and IEEE 802.3 encapsulation, the MSS value can be at most 1460

[1]The terms TCP segment and TCP packet are used interchangably

bytes and 1452 bytes respectively.

2.3 The Sliding Window Protocol

TCP imposes data flow control between the sender and the receiver and between the receiver's buffer and the receiver's receiving application by means of a sliding window protocol which is illustrated in figure 2.2. The gray area represents the sliding window whose size is initially equal

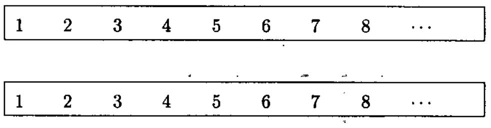


Figure 2.2: The sliding window protocol, $w_{\mathrm{rx}} = 4$.

to the advertised window size w_{rx} as determined by the receiver (4 packets in this case). w_{rx} is the maximum number of packets that the sender can transmit without having to wait for ACKs. For every ACK received, the window slides one packet forward and the next packet is transmitted. In figure 2.2, packets number 3 to 6 (4 in total) are sent after which the sender waits for ACKs. After a while, an ACK is received with sequence number 5, indicating that all data packets up to packet 4 have been received successfully and that the next packet to be received is packet 5. The sliding window moves two ahead and packets number 7 and 8 are sent. If for some reason, packet 6 arrives at the receiver before packet 5, the receiver will immediately respond with a duplicate ACK with sequence number 5. This indicates that a packet has arrived (packet 6 in this case), but that packet 5 is still outstanding.

Often, an ACK will be delayed in case data arrives that can be sent along with it. If data arrives, a data packet is generated and the ACK is sent along with it (sometimes referred to as *piggybacked*). Most TCP implementations will delay an ACK by up to 200 ms.

w_{rx} is determined at connection setup time, and is updated by the receiver with every ACK returned. The receiver uses w_{rx} to impose flow control between its buffer and its receiving application. It will, for example, reduce w_{rx} if the receiving application has not processed all the packets in its buffer. The sender uses w_{rx} to limit the rate at which packets are admitted into the network.

2.4 Average Round Trip Time Estimation

In order for TCP to implement certain of its congestion avoidance algorithms, it is necessary for it to estimate the average round trip time where the round trip time (RTT) is defined as the time from the start of a data packet's transmission until the time at which the corresponding ACK is received. An average for the RTT is obtained by updating a running average of the RTT with measurements obtained from the ACKs received.

The original algorithm (see Postal [67]) for RTT estimation updated the average RTT R and a value called the initial retransmission timeout value T_0 for every RTT measurement M as

$$R \leftarrow \alpha R + (1-\alpha)M \quad \text{and} \quad T_0 = \beta R$$

where α usually defaulted to 0.9 and β was recommended to have a value of 2. This algorithm proved to be inaccurate when large fluctuations in the RTT measurements occur and was replaced by Jacobson's algorithm [32]. Jacobson's algorithm updates R and T_0 with every RTT measurement M as

$$\begin{aligned} Err &= M - A \\ A &\leftarrow A + g \times Err \\ D &\leftarrow D + h(|Err| - D) \\ T_0 &= A + xD \end{aligned} \tag{1}$$

where $g = 1/8$ and $h = 1/4$. A and D are the running average (initialised to 0) and the mean deviation (initialised to 3) respectively of the RTT. The initial algorithm had $x = 2$ but was later changed (see Jacobson [33]) so that $x = 4$ except when T_0 is initialised in which case $x = 2$, resulting in $T_0 = 0 + 2 \times 3 = 6$ seconds.

Another value called the retransmission timeout value T_{RTO} depends on the value of T_0 as

$$T_{\mathrm{RTO}} = 2^{\beta} T_0$$

where 2^{β} is a multiplying factor which will be explained in a later section.

Karn's algorithm (see Karn and Partridge [35]) is additional to Jacobson's algorithm, and specifies that R, T_0 and T_{RTO} must not be updated from measurements of ACKs that acknowledge retransmitted packets.

2.5 Round Trip Time Measurements

TCP has a crude way of measuring the RTT. Not all packets that are sent and acknowledged are used for RTT measurements. A TCP connection has a timer that measures the RTT for one packet at a time and is initialised when that packet (the tagged packet) is transmitted. Every 500 msec a counter is incremented by one tick. When the ACK for the tagged packet arrives after say 550 msec, the RTT could be measured as either 1 tick (500 msec) or 2 ticks (1000 msec).

When the sequence number of the tagged packet is included in the ACK for another packet (due to the delayed ACK mechanism) the timer is turned off and the RTT measurement is declared void. Additionally, if the tagged packet had to be retransmitted the measurement is also declared void (a consequence of Karn's algorithm).

2.6 Bulk Data Transfers

TCP traffic consists mainly of two types, bulk data transfers (generated by *e.g.* FTP and SMTP) where the data payloads of the TCP packets tend to be full sized, and interactive data transfers (generated by *e.g.* Telnet and Rlogin) where the data payloads of the TCP packets are typically less than ten bytes. TCP handles both types of traffic, but it does so with different algorithms. This section discusses the algorithms that TCP uses for handling large data transfers.

2.6.1 Slow Start and Congestion Avoidance

Just as it is the responsibility of the receiver to manage flow control between its buffer and its receiving application, so is it the responsibility of the sender to adapt its flow of data into the network according to the resources available in the network. Instead of performing transmission by injecting w_{rx} packets as quickly as possible into the network which could cause congestion in the network, TCP uses two intelligent flow control algorithms called slow start (SS) and congestion avoidance (CA).

TCP makes use of two extra state variables to implement SS and CA: the congestion window size *cwnd* and the slow start threshold *ssthresh*. At the start of the connection *cwnd* is initialised to 1 packet and *ssthresh* to 65535 bytes. From this point onwards *ssthresh* is given in packets. The sender can always transmit up to the minimum of *cwnd* and w_{rx}.

TCP starts its transmission in SS where *cwnd* is increased by one packet for each ACK received, even when *cwnd* exceeds w_{rx}. This way of increasing the window results in an almost exponential window growth. (Its not exactly exponential as the receiver may delay its ACKs.) This implies that *cwnd* is incremented by one regardless of the number of packets acknowledged by an ACK. TCP will remain in SS until either a packet loss occurs or *cwnd* becomes larger than *ssthresh*. In the latter case, TCP exits SS and continues transmission with the CA algorithm.

The CA algorithm increments *cwnd* by $1/cwnd$ with every ACK received. This implies that *cwnd* increases by at most one packet per round trip time, leading to an additive increase of the congestion window as opposed to the exponential increase during SS. As in SS, *cwnd* is increased even when exceeding w_{rx}.

The values of *cwnd* and *ssthresh* dictate which algorithm is being performed. Whenever $cwnd \leq ssthresh$, SS is performed, and CA otherwise.

2.6.2 Packet Loss Detection and Retransmission

As TCP is responsible for the reliable transport of data, it must detect and respond to packet loss. For every packet that is transmitted, a timer is set to $T_{\text{RTO}} = 2^{\beta}T_0|_{\beta=0}$. If a packet is not acknowledged by the time its timer expires, a timeout (TO) occurs and the packet is assumed to be lost. A packet is also assumed to be lost when three duplicate ACKs are received, resulting in a triple duplicate ACK (TD) loss.

When a loss is detected, *ssthresh* is set to

$$\textit{ssthresh} = \max(2, \min(w_{\text{rx}}, \textit{cwnd})/2).$$

Further response by TCP depends on the type of loss.

Loss Due to TO

If the loss is a TO, T_{RTO} is set to $2^{\beta}T_0|_{\beta=1}$ seconds. The lost packet is retransmitted and the sender waits for the corresponding ACK to arrive. Each time that the timer expires, T_{RTO} is doubled up to a maximum of 64 seconds. This doubling is called exponential backoff. When the lost packet is finally acknowledged after one or more retransmissions, *cwnd* is reset to one and TCP proceeds in SS. If no packet can get through the network, TCP will eventually close the connection after 9 minutes (2 minutes for Solaris 2.x).

Karn's algorithm has the following implication. Say packet 1 is lost due to timeout, and after one or more retransmissions, it is successfully acknowledged with T_{RTO} set to $T_{\text{RTO}} = 2^{\beta}T_0$. Then packet 2 will be transmitted with its timer set to T_{RTO}. Only after a packet which was not retransmitted is acknowledged, are R, T_0 and T_{RTO} updated with Jacobson's algorithm.

Loss Due to TD

If a TD loss has occurred, TCP retransmits the missing packet without waiting for its timer to expire. This is the fast retransmit (FT) algorithm.

After retransmission, the fast recovery (FR) algorithm is performed: *cwnd* is set to *ssthresh* + 3 packets and incremented by one packet each time a duplicate ACK is received. The sender continues to transmit new packets but only up to the minimum of *cwnd* and w_{rx}. When finally an ACK is received that acknowledges new data, *cwnd* is set to *ssthresh* and TCP proceeds in the CA mode.

This process of increasing the congestion window linearly in CA mode and halving it after TD losses is called *additive increase and multiplicative decrease* (AIMD).

2.7 Interactive Data Transfers

This section briefly discusses TCP behaviour for interactive data transfers for applications such as Rlogin, where TCP packets are exchanged for every keystroke as illustrated in figure 2.3.

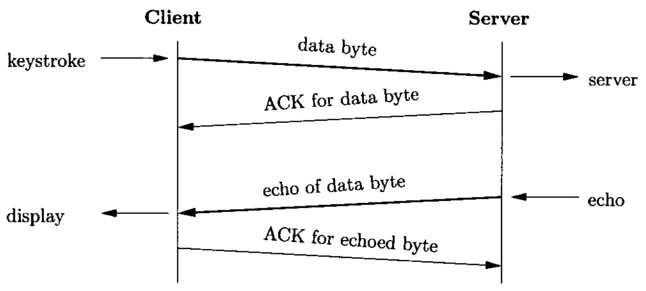


Figure 2.3: TCP packet transfers for single keystrokes in Rlogin.

When a keystroke occurs, one TCP packet with one byte in the data payload is sent from the client to the server. The server responds with an ACK for the byte of data after which it sends another packet to echo the data byte. Finally, the client acknowledges the echoed byte of data. Some applications, like Telnet, may send lines of input at a time.

A simple algorithm called the Nagle algorithm [54] is used to further reduce congestion. Instead of generating a TCP packet for every key stroke, data bytes are accumulated until an ACK is received for previously sent data, after which the new data are sent in one packet. In some cases, Nagle's algorithm is disabled when real-time data such as mouse movements need to be sent with as little delay as possible.

Packet loss is detected in the same way as for bulk transfers with either timeouts or duplicate ACKs. Lost packets are retransmitted and transmission proceeds as before.

2.8 TCP Variants

Despite the fact that TCP/IP is a well defined protocol suite, it comes in many variants as almost every operating system has its own implementation with updates and changes to every new system release. Most of these implementations were derived from the TCP/IP source code developed at the Computer Systems Research Group at the University of California at Berkeley and which was distributed with the 4.x BSD (Berkeley Software Distribution) systems and the *BSD Networking Releases*. Figure 2.4, taken from [75, Chapter 1], shows the various BSD releases in chronological order and indicates the relevant newly added TCP features.

In the rest of this thesis, we will use Tahoe and Reno to refer to the specific TCP implementations instead of the BSD releases. Table 2.2 shows the lineage of some TCP implementations.

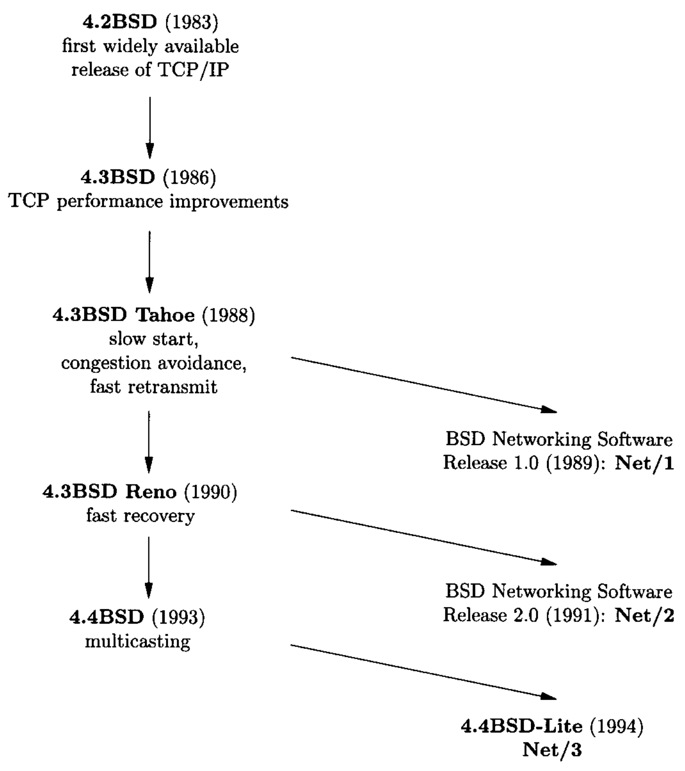


Figure 2.4: Various BSD releases with newly added TCP/IP features.

Except for operating systems with independent TCP/IP implementations (of which the source code is not publicly available), most TCP implementations are derived from either Tahoe or Reno but mostly from Reno (see Paxson [64, 65]).

2.8.1 TCP Tahoe

Tahoe implements SS, CA and FT, but not FR. Furthermore, SS is only performed if one end of the connection is on a different network. When a TD loss occurs, the lost packet is retransmitted and transmission proceeds in the SS mode.

2.8.2 TCP Reno

Reno implements most of the features discussed in section 2.6. It implements SS, CA and FT as well as FR. Unlike Tahoe, SS is always performed. Reno's increase ΔW of the congestion window during CA differs from that discussed earlier as it adds a fraction of the MSS in the increase

$$\Delta W = \frac{1}{cwnd} + \frac{1}{8\text{MSS}}.$$

Implementation	Tahoe	Reno	Independent
AIX		✓	
BSD/386		✓	
BSDI		✓	
DEC OSF/1		✓	
IRIX		✓	✓
Linux			✓
Microsoft Windows			✓
NetBSD		✓	
Solaris			✓
SunOS	✓		
Trumpet/Winsock			✓
Unix V/386			✓

Table 2.2: Lineage of some TCP implementations.

TCP Reno suffers from many deficiencies. The following are a few examples.

1. It fails to achieve fairness among multiple TCP connections that compete for bandwidth (see Mo *et al.* [53]). Connections with longer propagation delays typically receive less bandwidth.

2. Reno has an aggressive way of utilising the available bandwidth with its AIMD algorithm. This leads to oscillation in the window size and round trip time and also to high buffer occupancies (see Mo *et al.* [53]).

3. The RTT estimation scheme is too crude. Tests (see Brakmo *et al.* [9]) revealed that the retransmission timeout value T_{RTO} was on average 1100 msec long whereas the correct value would have been 300 msec if a more accurate clock had been used.

4. Reno suffers from severe throughput deficiencies (see Floyd and Fall [22] and Floyd [20]) when more than one packet is lost in one window of data. When this occurs, Reno loses its self-clocking as it cannot estimate the amount of data outstanding in the network.

   When multiple packets are lost in a window, and the first loss is indicated by a TD loss, TCP will execute the FT algorithm followed by FR as described in section 2.6.2. Therefore the first of the lost packets (the tagged packet) is retransmitted and if successful an ACK for the tagged packet will arrive which acknowledges some but not all of the packets transmitted prior to FT. This acknowledgment is called a partial ACK. In Reno, such a partial ACK will take TCP out of FR and consequently timeouts might occur for the other lost packets, especially if more than three packets were dropped.

   After the retransmissions caused by the timeouts, Reno proceeds with SS during which the throughput is also much lower than if TCP had continued with CA.

To address these deficiencies, Reno was extended in protocols such as TCP Lite, TCP SACK, TCP FACK, TCP New-Reno and TCP Vegas .

2.8.3 TCP Lite

Lite (see figure 2.4 and Stevens [76]) is a widely-used successor to Reno and is also known as Net/3. It provides, amongst others, support for transmission over links with high bandwidths and large propagation delays. The congestion control algorithms are essentially the same as for Reno.

2.8.4 TCP SACK

TCP SACK is basically the Reno protocol with two additional features called *selective acknowledgment* (SACK) and *selective retransmission* which are collectively referred to as the SACK mechanisms. The SACK mechanisms were introduced to deal specifically with point 4 in section 2.8.2 They were originally described by Jacobson and Braden [34] and was further modified by Mathis *et al.* [47].

With SACK, the receiver can inform the sender about all the data packets that have been successfully received. Therefore, if multiple packets are lost in a window, TCP selectively retransmits only the packets that have been lost and proceeds thereafter in congestion avoidance mode.

Simulation studies (see Fall and Floyd [22])) have shown that TCP SACK yields significantly better throughput than Tahoe and Reno.

2.8.5 TCP FACK

TCP FACK (see Mathis *et al.* [46]) extends TCP SACK with an algorithm called *Forward Acknowledgment* (FACK) which works in conjunction with SACK. SACK determines which data packets have to be retransmitted and FACK controls the injection (sending rate) of that data into the network by keeping an accurate estimate of the amount of data outstanding in the network by using the additional information provided by SACK.

2.8.6 TCP New-Reno

New-Reno (see Floyd [21]) is the Reno protocol with an important alteration to the FR algorithm to deal with point 4 in section 2.8.2 when the SACK option is not available.

The changes to the FR algorithm are as follows. When a partial ACK is received, the FR algorithm is not exited. The first unacknowledged packet is assumed to be lost and is retransmitted. Any subsequent duplicate ACKs will result in a FT, and New-Reno will exit FR only when an ACK arrives which acknowledges all data sent prior to the first retransmission in FT.

2.8.7 TCP Vegas

TCP Vegas (see Brakmo *et al.* [9, 10], Low *et al.* [41] and Mo *et al.* [53]) is a new implementation of TCP that in many ways is more sophisticated than TCP Reno, and tries to improve on the shortcomings of Reno. It is still in the development phase, and was designed to use available network resources more efficiently and fairly than TCP Reno. These goals are achieved by the following enhancements.

RTT Estimation

Instead of using the coarse-grained timer mechanism as described in section 2.5, Vegas records the system clock for each data packet upon transmission. When the corresponding ACK arrives, a more accurate RTT sample is calculated from the recorded time and the time on the system clock upon ACK arrival. This leads to more accurate RTT estimation, and enables Vegas to detect losses sooner than Reno.

Packet Loss

As Vegas detects losses more quickly than Reno, it only decreases the congestion window if the lost data packet was originally sent after the last window decrease. Any lost data packets that were transmitted before the window decrease do not indicate that the network is congested for the current congestion window size, and therefore do not imply that the window should be decreased again.

Congestion Avoidance

Vegas uses a different method for adjusting the congestion window during CA mode. It monitors two quantities, the expected packet sending rate R_{exp} and the actual packet sending rate R_{act}. R_{exp} approximates the rate at which data packets can be sent when the network is not congested and is equal to

$$R_{\text{exp}} = cwnd/RTT_{\text{base}}$$

where *cwnd* is the current congestion window size and RTT_{base} is the minimum RTT sample measured thus far. R_{act} is the actual sending rate of data packets and is equal to

$$R_{\text{act}} = cwnd/RTT_{\text{avg}}$$

where RTT_{avg} is the estimate of the average round trip time. The number of data packets B in the router buffers is then approximated as

$$B = (R_{\text{exp}} - R_{\text{act}})RTT_{\text{base}}.$$

Vegas updates *cwnd* every RTT based on the value of B as follows

$$cwnd = \begin{cases} cwnd + 1 & \text{if } B < \alpha \\ cwnd - 1 & \text{if } B > \beta \\ cwnd & \text{otherwise} \end{cases}$$

where α and β are respectively the minimum and maximum allowable number of data packets in the router buffers (α and β are parameters to TCP Vegas). Vegas thus uses the difference between the expected and actual data packet flow rates to estimate the available bandwidth in the network.

Slow Start

Vegas has minor modifications to SS. The congestion window is increased every other round so that valid comparisons between the actual and the expected data packet sending rates can be made when the congestion window is fixed. When the actual rate falls below the expected rate by an amount γ, Vegas exits SS and proceeds with CA. γ is also a parameter of TCP Vegas.

Issues Surrounding TCP Vegas

Vegas has issues that need to be resolved before it will gain support. The following are a few examples of which some have been addressed.

1. Vegas has problems when traffic is re-routed on other paths. For these paths the minimum RTT may be different than for the old path, and therefore Vegas will have to adjust the value of RTT_{base}. A method for doing this is presented in Mo *et al.* [53].
2. In Low *et al.* [41], a formal proof is given that multiple Vegas sources share bandwidth fairly, even when some have longer propagation delays. It is also shown that TCP Vegas connections do not interfere with TCP Reno connections. Vegas, however, does not receive fair bandwidth allocation when running alongside Reno.
3. Traffics from Vegas sources are easier to model than Reno. The work in Low *et al.* [41] focuses on the modelling of TCP Vegas traffic.

2.9 Modelling TCP

TCP is a complex protocol. Its behaviour is not memoryless, as its present behaviour depends on the history of the connection. As a result, TCP is difficult to model, and to create a model for every TCP implementation and for every network scenario such as the ones discussed in sections 5.1 and 6.1 is not realistic.

Most models focus on a specific network scenario and include the main features of TCP to obtain an average representation of the TCP traffic on that network. In general, connection setup and termination are not modelled while features like SS, CA, TO losses and TD losses are included. The aim of these models is to predict, among other performance metrics, the throughput per TCP connection.

In chapters 4, 5 and 6 we examine TCP models for a single, a finite number of and an infinite number of TCP sources respectively. (See appendix B for a list of TCP models that were not considered for this thesis.) We first present some mathematical background to the models.

Chapter 3

Mathematical Background

This chapter gives some background on some of the mathematical concepts that are used by the TCP models discussed in subsequent chapters. It begins with matrix notation and probability theory, followed by an introduction to phase type distributions, power-tail distributions and Markov modulated Poisson processes. Finally, two interesting queuing systems are discussed.

3.1 General Matrix Notation and Functions

A matrix with m rows and n columns is referred to as an $m \times n$ matrix. Matrices are denoted by bold uppercase letters *e.g.* $\mathbf{A}$. The ij-th component of the matrix $\mathbf{A}$ is denoted as A_{ij}. Vectors are denoted by bold lowercase letters *e.g.* $\mathbf{a}$. The i-th component of the vector $\mathbf{a}$ is referred to as a_i. Matrices and vectors may be written in a partitioned form such as

$$\mathbf{A} = \left(\begin{array}{c|c} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \hline \mathbf{A}_{21} & \mathbf{A}_{22} \end{array}\right) = \left(\begin{array}{cc} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{array}\right) \quad \text{and} \quad \mathbf{a} = \left(\begin{array}{c|c} \mathbf{a}_1 & \mathbf{a}_2 \end{array}\right) = \left(\begin{array}{cc} \mathbf{a}_1 & \mathbf{a}_2 \end{array}\right)$$

where $\mathbf{A}_{11}$, $\mathbf{A}_{12}$, $\mathbf{A}_{21}$ and $\mathbf{A}_{22}$ are matrices and $\mathbf{a_1}$ and $\mathbf{a_2}$ are vectors. If all the elements of $\mathbf{A}_{12}$ are equal to zero then

$$\mathbf{A} = \left(\begin{array}{c|c} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \hline \mathbf{A}_{21} & \mathbf{A}_{22} \end{array}\right) = \left(\begin{array}{c|c} \mathbf{A}_{11} & \\ \hline \mathbf{A}_{21} & \mathbf{A}_{22} \end{array}\right).$$

$\mathbf{e}$ and $\mathbf{0}$ denote vectors whose elements are all equal to 1 and 0 respectively and with appropriate dimensions depending on where they are used in matrix equations. The matrix $\mathbf{I}$ is the identity matrix.

The natural number e to the power $\mathbf{A}x$ where $\mathbf{A}$ is a square matrix and x is a scalar is defined as

$$e^{\mathbf{A}x} = \exp(\mathbf{A}x) = \sum_{i=0}^{\infty} \frac{(\mathbf{A}x)^i}{i!}.$$

In the rest of this thesis e, e_r and $e_{r'}$ denote loss probabilities and should not be confused with the natural number e or the vector $\mathbf{e}$. The derivative of $\exp(\mathbf{A}x)$ with respect to x is

$$\frac{d\exp(\mathbf{A}x)}{dx} = \mathbf{A}\exp(\mathbf{A}x).$$

The Kronecker product of two matrices $\mathbf{L}$ and $\mathbf{M}$ is defined as follows (see Neuts [56, Chapter 2]). Let $\mathbf{L}$ and $\mathbf{M}$ be rectangular matrices of dimensions $m_{\text{L}} \times n_{\text{L}}$ and $m_{\text{M}} \times n_{\text{M}}$. The Kronecker product $\mathbf{L} \otimes \mathbf{M}$ is the matrix of dimension $m_{\text{L}}m_{\text{M}} \times n_{\text{L}}n_{\text{M}}$, written in block-partitioned form as

$$\begin{pmatrix} L_{11}M & L_{12}M & \dots & L_{1n_{\text{L}}}M \\ \vdots & \vdots & & \vdots \\ L_{m_{\text{L}}1}M & L_{m_{\text{L}}2}M & \dots & L_{m_{\text{L}}n_{\text{L}}}M \end{pmatrix}.$$

The Kronecker sum $\mathbf{L} \oplus \mathbf{M}$ (see Schwefel [71]) is

$$\mathbf{L} \oplus \mathbf{M} = \mathbf{L} \otimes \mathbf{I}_{m_{\text{M}} \times m_{\text{M}}} + \mathbf{I}_{m_{\text{L}} \times m_{\text{L}}} \otimes \mathbf{M}$$

where $\mathbf{L}$ is either a square matrix or a column vector. Further define

$$\mathbf{A}^{\otimes n} = \mathbf{A} \otimes \cdots \otimes \mathbf{A} \quad \text{(n times)},$$
$$\mathbf{A}^{\oplus n} = \mathbf{A} \oplus \cdots \oplus \mathbf{A} \quad \text{(n times)}$$

where

$$\mathbf{A}^{\otimes 1} = \mathbf{A} \quad \text{and} \quad \mathbf{A}^{\oplus 1} = \mathbf{A}.$$

Finally, define the following functions for any matrix $\mathbf{A}$

rows($\mathbf{A}$)	Number of rows of $\mathbf{A}$
col($\mathbf{A}$)	Number of columns of $\mathbf{A}$
$\lvert\mathbf{A}\rvert$	rows($\mathbf{A}$) $\times$ col($\mathbf{A}$)

3.2 General Probability Theory

Let $P(A)$ denote the probability for an event $A \in S$ where S is the sample space and

$$0 \le P(A) \le 1 \quad \text{and} \quad P(S) = 1.$$

Let X be a continuous random variable. Then X has a cumulative probability distribution function (CDF) $F(x)$ defined as

$$F(x) = P(X \le x).$$

$F(x)$ is often referred to as the distribution function of X. The reliability function $R(x)$ of X is defined as

$$R(x) = P(X > x) = 1 - F(x)$$

and the probability density function (pdf) $f(x)$ of X, if it exists, is defined as

$$f(x) = \frac{dF(x)}{dx} = -\frac{dR(x)}{dx}.$$

The expectation $\mathrm{E}(X^i)$ of X^i, also referred to as the i-th moment of X, if it exists, is defined as

$$\mathrm{E}(X^i) = \int_{-\infty}^{\infty} x^i f(x)dx = \int_{-\infty}^{\infty} x^i dF.$$

Alternatively, the i-th moment can also be computed with the help of the Laplace-Stieltjes transform $L(s)$ of X when $f(x) = F(x) = 0$ for $x < 0$:

$$\mathrm{E}(X^i) = (-1)^i \left.\frac{d^i L(s)}{ds^i}\right|_{s=0}$$

where

$$\begin{aligned} L(s) &= \int_0^{\infty} \exp(-sx)dF(x) \\ &= \int_0^{\infty} \exp(-sx)f(x)dx \quad (\text{if } f(x) \text{ exists}). \end{aligned}$$

The variance of X, if it exists, is given by

$$\mathrm{Var}(X) = \sigma^2 = \mathrm{E}((X - \mathrm{E}(X))^2) = \mathrm{E}(X^2) - (\mathrm{E}(X))^2.$$

3.3 Phase Type Distributions

Phase type (PH) distributions or matrix exponential distributions are defined in Latouche and Ramaswami [25, Chapter 2], Lipsky [55] and Neuts [56, Chapter 2]. The description here follows the presentation in Neuts.

Consider a continuous-time Markov chain (CTMC) on the states $\{1, 2, \ldots, m+1\}$ with infinitesimal generator

$$\mathbf{Q} = \begin{pmatrix} \mathbf{T} & \mathbf{t} \\ \mathbf{0} & 0 \end{pmatrix}$$

where $\mathbf{T}$ is an $m \times m$ matrix that contains the non-absorbing states (states 1 to m) of the CTMC and satisfies $T_{ii} < 0$ for $1 \leq i \leq m$ and $T_{ij} \geq 0$ for $i \neq j$. The column vector $\mathbf{t}$ is such that $\mathbf{Te} + \mathbf{t} = \mathbf{0}$ and the initial probability vector of $\mathbf{Q}$ is given by $(\mathbf{a}, a_{m+1})$ with $\mathbf{ae} + a_{m+1} = 1$.

It is assumed that the first m states are transient so that absorption into state $m + 1$, from any initial state, is guaranteed. The distribution function $F(\cdot)$ on $[0, \infty)$ of the time until absorption in state $m + 1$, with initial probability vector $(\mathbf{a}, a_{m+1})$, is given by

$$F(x) = 1 - \mathbf{a}\exp(\mathbf{T}x)\mathbf{e}, \quad \text{for} \quad x \geq 0.$$

The distribution $F(\cdot)$ is a distribution of phase type and the pair $< \mathbf{a}, \mathbf{T} >$ is the PH representation of $F(\cdot)$. The probability density function $f(x)$ is

$$f(x) = \mathbf{a}\exp(\mathbf{T}x)\mathbf{t}$$

which has a Laplace-Stieltjes transform

$$L(s) = a_{m+1} + \mathbf{a}(s\mathbf{I} - \mathbf{T})^{-1}\mathbf{t}, \quad s \geq 0.$$

The i-th moment $\mathrm{E}(x^i)$ of $< \mathbf{a}, \mathbf{T} >$ is given by

$$\mathrm{E}(x^i) = \frac{d}{dx}L(s)\bigg|_{s=0} = (-1)^i i!(\mathbf{a}\mathbf{T}^{-i}\mathbf{e}), \quad i \geq 0.$$

3.4 The Power-Tail Distribution

This section follows the presentation in Schwefel [71]. A finite mixture of exponential distributions has a reliability function $R(x) \sim \exp(-x)$ that drops off exponentially for $x > y$. In contrast, the reliability function of a power-tail (PT) distribution drops off by a power of x, which is slower than that of the exponential:

$$R(x) \to \frac{c}{x^\alpha} \text{ for large } x.$$

This is illustrated in figure 3.1 which plots the Pareto PT and the exponential distribution reliability functions over two intervals.

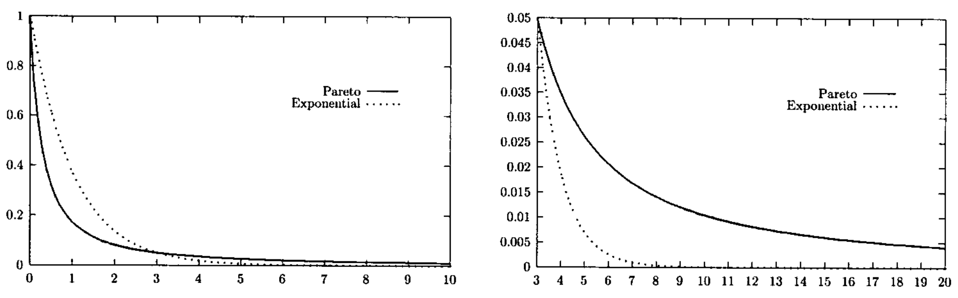


Figure 3.1: The reliability functions of the Pareto and the exponential distributions with unit means.

α is the shape parameter. A PT distribution has the added characteristic that if $\alpha \leq 2$ it has an infinite variance, and if $\alpha \leq 1$ it has an infinite mean.

It is known that Internet traffic, which includes TCP traffic, is bursty, and the Pareto PT distribution has become almost synonymous with the modelling of such traffic (see Crovella and Bestavros [16], Greiner *et al.* [27], Leland *et al.* [39] and Paxson and Floyd [66]). For modelling these traffics a typical value for α is chosen at around 1.4 (see [16, 39]).

The Pareto distribution with mean U has a distribution function

$$F(x) = 1 - \frac{1}{\left(\frac{x}{(\alpha-1)U} + 1\right)^\alpha}. \tag{2}$$

PT distributions do not have exact PH representations, but can be approximated by truncated power-tail (TPT) distributions which asymptotically have PT characteristics. We consider a hyper-exponential TPT distribution $F_m(x)$ with m phases

$$F_m(x) = \sum_{i=1}^{m} a_i(1 - \exp(-r_i x))$$

where the entrance probabilities a_i and the state leaving rates r_i are

$$a_i = \frac{\theta^{i-1}(1-\theta)}{1-\theta^m}, \quad r_i = \frac{\mu}{\gamma^{i-1}}$$

and where $0 < \theta < 1$ is usually set to 0.5 and $\gamma > 1$. $F_m(x)$ has a PH representation $< \mathbf{p}, \mathbf{T} >$ where $\mathbf{p} = \{a_1, \ldots, a_m\}$ and

$$\mathbf{T} = \begin{pmatrix} -r_1 & & \\ & \ddots & \\ & & -r_m \end{pmatrix}.$$

The reliability function $R_m(x)$ of the TPT is

$$R_m(x) = \frac{1-\theta}{1-\theta^m} \sum_{i=1}^{m} \theta^{i-1} \exp(-r_i x).$$

When $m \to \infty$ the distribution has a PT with $\alpha = -\log(\theta)/\log(\gamma)$. Given a value for α, we need to set γ to $\gamma = \theta^{-1/\alpha}$. In order for the TPT distribution to have an expected value of U, we set μ to

$$\mu = \frac{1-\theta}{1-\theta^m} \frac{1-(\theta\gamma)^m}{1-\theta\gamma} \frac{1}{U}.$$

3.5 Markov Modulated Poisson Process

This section follows the presentation in Lipsky and Fiorini [40, Section 3.5] and Schwefel [72, Appendix D.2]. A Markov modulated Poisson process (MMPP) forms part of a class of stochastic processes called semi-Markov (SM) processes. A MMPP is used to describe an arrival process where the Poisson arrival rate is modulated (changed) according to a Markov chain.

Let $\mathbf{P}$ be the transition matrix for a Markov chain with m states, and let $1/\mu_i$ be the mean time that the chain spends in state i. Suppose that when the Markov chain is in state i, packets are emitted with Poisson rate λ_i. Let $\mathbf{M}$ and $\mathbf{L}$ be the diagonal matrices

$$\mathbf{M} = \begin{pmatrix} \mu_1 & & \\ & \ddots & \\ & & \mu_m \end{pmatrix} \quad \text{and} \quad \mathbf{L} = \begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_m \end{pmatrix}.$$

The infinitesimal generator $\mathbf{Q}$ for the modulating process is defined as $\mathbf{Q} = \mathbf{M}(\mathbf{I} - \mathbf{P})$. The stationary distribution $\boldsymbol{\pi}$ of the process has the property

$$\boldsymbol{\pi}\mathbf{Q} = \mathbf{0}$$

where π_i is the probability that the modulating process is in state i at an arbitrary point in time. The average arrival rate λ_{avg} is

$$\lambda_{\text{avg}} = \sum_{i=1}^{m} \pi_i \lambda_i. \tag{3}$$

The use of the matrices $\mathbf{Q}$ and $\mathbf{L}$ and the vector $\boldsymbol{\pi}$ will become apparent in the next sections. Next we look at examples of MMPPs.

3.6 MMPP 1-Burst: A Single ON/OFF Source

1-Burst (see Schwefel [71, 72]) is a MMPP which models an ON/OFF traffic source which transmits packets at a rate of $\lambda_{\text{tot}} = \lambda_{\text{low}} + \lambda_{\text{ON}}$ (packets/s) during ON periods and at a rate of λ_{low} (packets/s) during OFF periods. λ_{low} accounts for low background traffic and λ_{ON} for the burstiness of the traffic being modelled.

The packet streams during the ON and OFF periods are assumed to be Poisson. OFF periods are exponentially distributed with mean Z and ON periods have a PH distribution with representation $< \mathbf{p}, \mathbf{T} >$ and mean U.

The infinitesimal generator matrix $\mathbf{Q}_1$ and the matrix $\mathbf{L}_1$ which contains the Poisson rates for the MMPP are

$$\mathbf{Q}_1 = \left(\begin{array}{c|c} -1/Z & (1/Z)\mathbf{p} \\ \hline -\alpha\mathbf{Te} & \alpha\mathbf{T} \end{array}\right) \quad \text{and} \quad \mathbf{L}_1 = \left(\begin{array}{c|c} \alpha\lambda_{\text{low}} & \\ \hline & \alpha(\lambda_{\text{low}} + \lambda_{\text{ON}})\mathbf{I} \end{array}\right)$$

where α is a throttling factor that will be used in section 5.3 and is set to one by default. The average packet arrival rate $\lambda^{(1)}_{\text{avg}}$ is

$$\lambda^{(1)}_{\text{avg}} = \lambda_{\text{low}} + \lambda_{\text{ON}} \frac{U/\alpha}{Z + U/\alpha}. \tag{4}$$

For the special case where ON periods have a TPT distribution of order m (see section 3.4), the modulating process within the 1-Burst is illustrated in figure 3.2.

The process leaves the OFF state at rate $1/Z$ and goes to state i with probability a_i. When in state i, the process returns to the OFF state at rate r_i. The vector $\boldsymbol{\pi}$ (see section 3.5) which satisfies $\boldsymbol{\pi}\mathbf{Q}_1 = \mathbf{0}$ when ON periods have a TPT distribution is

$$\boldsymbol{\pi} = \{-1, \frac{\mathbf{p}_1}{\alpha Z \mathbf{T}_{1,1}}, \cdots, \frac{\mathbf{p}_m}{\alpha Z \mathbf{T}_{m,m}}\} = \{1, \frac{a_1}{\alpha Z r_1}, \cdots, \frac{a_m}{\alpha Z r_m}\}.$$

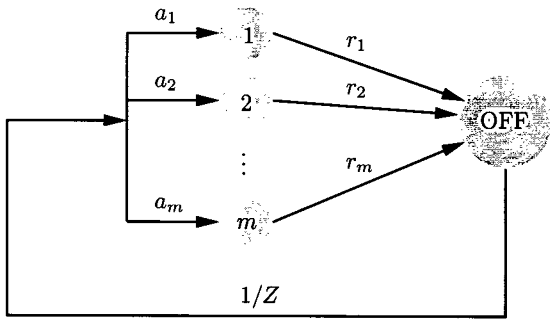


Figure 3.2: The modulating process within the 1-Burst process for ON periods with a TPT distribution.

3.7 MMPP N-Burst: N Aggregated 1-Burst Sources

Recall the parameters, vectors and matrices from sections 3.5 and 3.6. The N-Burst process (see Schwefel [71, Section 3.3] and Schwefel [72, Appendix E.1.2]) is an aggregation of N 1-Burst processes and can be represented by a MMPP with generator matrix $\mathbf{Q}_N = \mathbf{Q}_1^{\oplus N}$ and diagonal matrix $\mathbf{L}_N = \mathbf{L}_1^{\oplus N}$. The average packet arrival rate λ_{avg} is

$$\lambda_{\text{avg}} = N\lambda_{\text{avg}}^{(1)}. \tag{5}$$

Let $\varepsilon = \lambda_{\text{low}}/\lambda_{\text{ON}}$ so that

$$\lambda_{\text{avg}}^{(1)} = \lambda_{\text{ON}}\left(\varepsilon + \frac{U/\alpha}{Z + U/\alpha}\right).$$

In order for the N-Burst process to have an average packet arrival rate of λ_{avg}, we set

$$\lambda_{\text{ON}} = \frac{\lambda_{\text{avg}}}{N\left(\varepsilon + \frac{U}{Z+U}\right)}. \tag{6}$$

3.8 MMPP Modified N-Burst

The modified N-Burst Process (see Schwefel [72, Appendix E.1.2] and Schwefel [73, Appendix C]) is a more general form of the N-Burst process of section 3.7 and is used by the TCP-NBurst model in section 5.3.

For reasons that will become apparent in section 5.3, in the modified N-Burst process the throttling factor α depends on the number of active sources (the number of sources in the ON state). Let α_i denote the value of the throttling factor when i sources are active.

The modified N-Burst process is a MMPP where the infinitesimal generator matrix $\mathbf{Q}_N$ has a quasi-birth-and-death (QBD) structure with the matrix rows defined by the number of active sources

$$\mathbf{Q}_N = \left(\begin{array}{c|c|c|c|c} \mathbf{Z}_0 & \mathbf{X}_0 & & & \\ \hline \mathbf{Y}_1 & \mathbf{Z}_1 & \mathbf{X}_1 & & \\ \hline & \ddots & \ddots & \ddots & \\ \hline & & \mathbf{Y}_{N-1} & \mathbf{Z}_{N-1} & \mathbf{X}_{N-1} \\ \hline & & & \mathbf{Y}_N & \mathbf{Z}_N \end{array}\right)$$

where

$$\begin{array}{lll} \mathbf{Z}_0 & = -1/Z, & \\ \mathbf{X}_0 & = (1/Z)\mathbf{p}, & \\ \mathbf{X}_i & = \frac{N-i}{Z}\mathbf{I}^{\otimes i} \otimes \mathbf{p} & i = 1, \ldots, N-1, \\ \mathbf{Y}_i & = \alpha_i (\mathbf{Te})^{\oplus i} & i = 1, \ldots, N, \\ \mathbf{Z}_i & = -\alpha_i \mathbf{T}^{\oplus i} - \frac{N-i}{Z}\mathbf{I}^{\otimes i} & i = 1, \ldots, N. \end{array}$$

QBD processes are discussed later in this chapter. The corresponding diagonal matrix $\mathbf{L}_N$ containing the Poisson arrival rates is

$$\mathbf{L}_N = \left(\begin{array}{c|c|c|c|c} \lambda_{\text{low}} & & & & \\ \hline & \alpha_1\lambda_{\text{tot}}\mathbf{I} & & & \\ \hline & & \alpha_2(2\lambda_{\text{tot}})\mathbf{I}^{\otimes 2} & & \\ \hline & & & \ddots & \\ \hline & & & & \alpha_N(N\lambda_{\text{tot}})\mathbf{I}^{\otimes N} \end{array}\right).$$

The matrix $\mathbf{I}$ has the same dimensions as the matrix $\mathbf{T}$. Let λ_i and s_i denote respectively the aggregate Poisson arrival rate and number of active sources when the process is in state i, with π_i the steady state probability of being in that state. The average packet arrival rate $\lambda^{(1)}_{\text{avg}}$ per source is

$$\lambda^{(1)}_{\text{avg}} = \sum_{\text{all states } i} \pi_i \lambda_i / s_i.$$

3.9 MMPP N-Exp: N Aggregated ON/OFF Sources

The N-Exp process is a simplified version of the modified N-Burst process and is used by the TCP-AK1ext and TCP-AK2ext models in section 6.4. It is a MMPP which models N ON/OFF sources where both the OFF and ON periods are exponentially distributed. Let Z and U be the means of the OFF and ON periods respectively. Each source transmits packets at a rate of $\lambda_{\text{tot}} = \lambda_{\text{low}} + \lambda_{\text{ON}}$ (packets/s) during ON periods and at a rate of λ_{low} (packets/s) during OFF periods. Let $\varepsilon = \lambda_{\text{low}}/\lambda_{\text{ON}}$.

N-Exp has an infinitesimal generator matrix $\mathbf{Q}_N$ which has a QBD structure similar to that of the modified N-Burst process where the matrix rows are defined by the number of active sources

$$\mathbf{Q}_N = \left(\begin{array}{c|c|c|c|c|c} -\frac{N}{Z} & \frac{N}{Z} & & & & \\ \hline \frac{1}{U} & -(\frac{1}{U}+\frac{N-1}{Z}) & \frac{N-1}{Z} & & & \\ \hline & \frac{2}{U} & -(\frac{2}{U}+\frac{N-2}{Z}) & \frac{N-2}{Z} & & \\ \hline & & \ddots & \ddots & \ddots & \\ \hline & & & \frac{N-1}{U} & -(\frac{N-1}{U}+\frac{1}{Z}) & \frac{1}{Z} \\ \hline & & & & \frac{N}{U} & -\frac{N}{U} \end{array}\right). \tag{7}$$

The corresponding rate matrix $\mathbf{L}_N$ is

$$\mathbf{L}_N = \left(\begin{array}{c|c|c|c|c} \lambda_{\text{low}} & & & & \\ \hline & \alpha_1\lambda_{\text{tot}} & & & \\ \hline & & \alpha_2(2\lambda_{\text{tot}}) & & \\ \hline & & & \ddots & \\ \hline & & & & \alpha_N(N\lambda_{\text{tot}}) \end{array}\right).$$

The average arrival rate $\lambda_{\text{avg}}^{(1)}$ per source is

$$\lambda_{\text{avg}}^{(1)} = \lambda_{\text{low}} + \sum_{i=1}^{N} \pi_{i+1}\alpha_i\lambda_{\text{ON}} \tag{8}$$

where $\boldsymbol{\pi}$ is the vector that satisfies $\boldsymbol{\pi}\mathbf{Q} = \mathbf{0}$. The total average arrival rate λ_{avg} is

$$\lambda_{\text{avg}} = N\lambda_{\text{avg}}^{(1)}.$$

3.10 Quasi-Birth-and-Death Processes

We often wish to have a representation of a queue which has a more complex behaviour than for example the simple M/M/*/ * /* type queues. These queues may have more interesting arrival processes such as a MMPP and more realistic packet service time distributions with PH representations. We can model such a queue with a quasi-birth-and-death (QBD) process.

A QBD process (see Neuts [56, Chapter 3]) is a Markov process defined on the state space $E = \{(i,j) \mid i \geq 0,\ 1 \leq j \leq m\}$ with infinitesimal generator matrix $\mathcal{Q}$ given by

$$\mathcal{Q} = \begin{pmatrix} \mathbf{B}_0 & \mathbf{A}_0 & & & \\ \mathbf{B}_1 & \mathbf{A}_1 & \mathbf{A}_0 & & \\ & \mathbf{A}_2 & \mathbf{A}_1 & \mathbf{A}_0 & \\ & & \mathbf{A}_2 & \mathbf{A}_1 & \mathbf{A}_0 \\ & & & \ddots & \ddots & \ddots \end{pmatrix}$$

where $(\mathbf{B}_0+\mathbf{A}_0)\mathrm{e} = (\mathbf{B}_1+\mathbf{A}_1+\mathbf{A}_0)\mathrm{e} = (\mathbf{A}_0+\mathbf{A}_1+\mathbf{A}_2)\mathrm{e} = \mathbf{0}$. Typically, the matrix $\mathbf{A}_0$ represents the arrival process and the matrix $\mathbf{A}_2$ the service process. Two examples of QBD processes are the SM/M/1/K queue and a variant of the SM/M/1 queue which are discussed in the following two sections.

3.11 The SM/M/1/K Queue

This section follows the presentation in Schwefel [72, Appendix D.6] and discusses the SM/M/1/K queue where the arrival process is a MMPP, for example the 1-Burst arrival process, and where the service times are exponentially distributed with parameter μ. The SM/M/1/K queue is used by the TCP-AK1ext and TCP-AK2ext models in section 6.4.

Recall from section 3.5 the MMPP matrices $\mathbf{Q}$ and $\mathbf{L}$ and the vector $\boldsymbol{\pi}$. The SM/M/1/K queue can be analysed as a QBD process where the infinitesimal generator matrix $\mathcal{Q}$ has dimension $(K+1) \times \text{rows}(\mathbf{Q})$ and is given by

$$\mathcal{Q} = \begin{pmatrix} \overline{\mathbf{A}}_1 & \mathbf{A}_0 & & & & \\ \mathbf{A}_2 & \mathbf{A}_1 & \mathbf{A}_0 & & & \\ & \mathbf{A}_2 & \mathbf{A}_1 & \mathbf{A}_0 & & \\ & & \ddots & \ddots & \ddots & \\ & & & \mathbf{A}_2 & \mathbf{A}_1 & \mathbf{A}_0 \\ & & & & \mathbf{A}_2 & \underline{\mathbf{A}}_1 \end{pmatrix}$$

where

$$\overline{\mathbf{A}}_1 = -(\mathbf{L} - \mathbf{Q}), \quad \underline{\mathbf{A}}_1 = -(\mu\mathbf{I} - \mathbf{Q}),$$
$$\mathbf{A}_0 = \mathbf{L}, \quad \mathbf{A}_1 = -(\mathbf{L} + \mu\mathbf{I} - \mathbf{Q}), \quad \mathbf{A}_2 = \mu\mathbf{I}.$$

The matrix rows of $\mathcal{Q}$ are defined by the number of packets in the system. The stationary distribution $\{\mathbf{x}_i\}$ of the number i of packets in the system is

$$\mathbf{x}_i = \mathbf{a}\mathbf{R}^i + \mathbf{b}\mathbf{S}^{K-i} \quad i = 0, \ldots, K$$

where the matrices $\mathbf{R}$ and $\mathbf{S}$ are the minimal nonnegative solutions to the following quadratic matrix equations

$$\mathbf{A}_0 + \mathbf{R}\mathbf{A}_1 + \mathbf{R}^2\mathbf{A}_2 = \mathbf{0}, \quad \mathbf{S}^2\mathbf{A}_0 + \mathbf{S}\mathbf{A}_1 + \mathbf{A}_2 = \mathbf{0}.$$

Numerical methods to compute $\mathbf{R}$ and $\mathbf{S}$ are described in Krieger *et al.* [36] and Schwefel [72, Appendix F.1]. The coefficient row vectors $\mathbf{a}$ and $\mathbf{b}$ are the solutions to the following system of linear equations derived from the boundary equations $\mathbf{x}\mathcal{Q} = \mathbf{0}$

$$(\mathbf{a}|\mathbf{b}) \left(\begin{array}{c|c} \mathbf{A}_1 + \mathbf{A}_2 + \mathbf{R}\mathbf{A}_2 & \mathbf{R}^K(\mathbf{A}_0 - \mathbf{R}\mathbf{A}_2) \\ \hline \mathbf{S}^K(\mathbf{A}_2 - \mathbf{S}\mathbf{A}_0) & \mathbf{S}\mathbf{A}_0 + \mathbf{A}_1 + \mathbf{A}_0 \end{array} \right) = \mathbf{0}$$

and from the normalisation of $\mathbf{x}_i$.

The components of the vector $\mathbf{x}_i = \{(i,1), (i,2), \ldots, (i,m)\}$ are the probabilities that there are i packets in the system and that the arrival process is in the corresponding state (either one of states 1 to m where $m = \text{rows}(\mathbf{Q})$). The probabilities $\{a_i\}$ at arrival times are obtainable by scaling $\mathbf{x}_i$ by $\mathbf{L}$ as states of the arrival process with higher arrival rates contribute more arrival observation points

$$a_i = \frac{(\mathbf{a}\mathbf{R}^i + \mathbf{b}\mathbf{S}^{K-i})\mathbf{L}\mathbf{e}}{\boldsymbol{\pi}\mathbf{L}\mathbf{e}}, \quad i = 0, \ldots, K. \tag{9}$$

The denominator in equation (9) normalises $\{a_i\}$. The average queue length q is

$$\begin{aligned} q &= \sum_{i=1}^{K} i\mathbf{x}_i\mathbf{e} \\ &= \mathbf{a}(\mathbf{I}-\mathbf{R})^{-1}[(\mathbf{I}-\mathbf{R})^{-1}(\mathbf{I}-\mathbf{R}^{K+2}) - (K+1)\mathbf{R}^{K+1} - \mathbf{I}]\mathbf{e} \\ &\quad - \mathbf{b}(\mathbf{I}-\widetilde{\mathbf{S}})^{-1}[\mathbf{b}(\mathbf{I}-\widetilde{\mathbf{S}})^{-1}(\mathbf{I}-\mathbf{S}^{K+1}) - (K+1)\mathbf{I}]\mathbf{e} \end{aligned} \tag{10}$$

and the packet loss probability e is

$$e = a_K. \tag{11}$$

3.12 An SM/M/1 Queue With An Alternating MMPP

This section describes a QBD process where the arrival process alternates between two different MMPPs according to the number of packets in the system. It is the same process as described by Schwefel [73, Appendix D] and is used by the TCP-NBurst model in section 5.3.

Let MMPP number one and MMPP number two be described by the infinitesimal generator matrices $\mathbf{Q}$ and $\widetilde{\mathbf{Q}}$ and diagonal matrices $\mathbf{L}$ and $\widetilde{\mathbf{L}}$ respectively. When the number of packets in the system is less than B, arrivals will be according to MMPP number one; and when the number of packets in the system is B or more, the arrival process will be according to MMPP number two. Packet service times are exponentially distributed with mean $1/\mu$. The infinitesimal generator matrix $\mathcal{Q}$ of the QBD process is given by

$$\mathcal{Q} = \left(\begin{array}{cccccc} \overline{\mathbf{A}}_1 & \mathbf{A}_0 & & & & \\ \mathbf{A}_2 & \mathbf{A}_1 & \mathbf{A}_0 & & & \\ & \ddots & \ddots & \ddots & & \\ & & \mathbf{A}_2 & \mathbf{A}_1 & \mathbf{A}_0 & \\ \hline & & & \mathbf{C}_2 & \mathbf{C}_1 & \mathbf{C}_0 \\ & & & & \ddots & \ddots \quad \ddots \end{array} \right)$$

where

$$\begin{aligned} \overline{\mathbf{A}}_1 = -(\mathbf{L}-\mathbf{Q}), \quad \mathbf{A}_0 = \mathbf{L}, \quad \mathbf{A}_1 = -(\mathbf{L}+\mu\mathbf{I}-\mathbf{Q}), \quad \mathbf{A}_0 = \mu\mathbf{I} \\ \mathbf{C}_0 = \widetilde{\mathbf{L}}, \quad \mathbf{C}_1 = -(\widetilde{\mathbf{L}}+\mu\mathbf{I}-\widetilde{\mathbf{Q}}), \quad \mathbf{C}_0 = \mu\mathbf{I}. \end{aligned}$$

The steady state distribution of the number k of packets in the queue is

$$\mathbf{x}_k = \begin{cases} \mathbf{a}\mathbf{R}^k + \mathbf{b}\mathbf{S}^{B-1-k} & k = 0, \dots, B-1 \\ \mathbf{x}_B\mathbf{T}^{k-B} & k = B, B+1, \dots \end{cases}.$$

The matrix factors $\mathbf{R}$, $\mathbf{S}$ and $\mathbf{T}$ are the minimal nonnegative solutions of the quadratic matrix equations

$$\mathbf{A}_0 + \mathbf{R}\mathbf{A}_1 + \mathbf{R}^2\mathbf{A}_2 = \mathbf{0}, \quad \mathbf{A}_2 + \mathbf{S}\mathbf{A}_1 + \mathbf{S}^2\mathbf{A}_0 = \mathbf{0}, \quad \mathbf{C}_0 + \mathbf{T}\mathbf{C}_1 + \mathbf{T}^2\mathbf{C}_2 = \mathbf{0}\ . \tag{12}$$

Numerical methods to compute $\mathbf{R}$ and $\mathbf{S}$ are described in Krieger *et al.* [36] and Schwefel [72, Appendix F.1]. The vectors $\mathbf{a}$ and $\mathbf{b}$ follow from normalisation $\mathbf{xe} = 1$ as the solution to the system of linear equations

$$(\mathbf{a}|\mathbf{b})\left(\begin{array}{c|c|c} \mathbf{L}_a & \mathbf{R}_a & \mathbf{d}_1 \\ \hline \mathbf{L}_b & \mathbf{R}_b & \mathbf{d}_2 \end{array}\right) = (\mathbf{0},\mathbf{0},1) \tag{13}$$

where

$$\begin{aligned}
\mathbf{L}_a &= \overline{\mathbf{A}}_1 + \mathbf{R}\mathbf{A}_2 \\
\mathbf{L}_b &= \mathbf{S}^{B-2}(\mathbf{S}\overline{\mathbf{A}}_1 + \mathbf{A}_2) \\
\mathbf{R}_a &= \mathbf{R}^{B-2}(\mathbf{R}\mathbf{A}_0 - (\mathbf{A}_0 + \mathbf{R}\mathbf{A}_1)\mathbf{C}_2^{-1}(\mathbf{C}_1 + \mathbf{T}\mathbf{C}_2)) \\
\mathbf{R}_b &= \mathbf{A}_0 - (\mathbf{S}\mathbf{A}_0 + \mathbf{A}_1)\mathbf{C}_2^{-1}(\mathbf{C}_1 + \mathbf{T}\mathbf{C}_2) \\
\mathbf{d}_1 &= \sum_{k=0}^{B-1} \mathbf{R}^k\mathbf{e} - \mathbf{R}^{B-2}(\mathbf{A}_0 + \mathbf{R}\mathbf{A}_1)\mathbf{C}_2^{-1}\left(\sum_{\ell=0}^{\infty} \mathbf{T}^\ell\mathbf{e}\right) \\
\mathbf{d}_2 &= \sum_{k=0}^{B-1} \mathbf{S}^k\mathbf{e} - (\mathbf{S}\mathbf{A}_0 + \mathbf{A}_1)\mathbf{C}_2^{-1}\left(\sum_{\ell=0}^{\infty} \mathbf{T}^\ell\mathbf{e}\right).
\end{aligned}$$

Finally

$$\mathbf{x}_B = -\mathbf{a}\mathbf{R}^{B-2}(\mathbf{A}_0 + \mathbf{R}\mathbf{A}_1) + \mathbf{b}(\mathbf{S}\mathbf{A}_0 + \mathbf{A}_1)\mathbf{C}_2^{-1}.$$

As in the previous section, to obtain the steady state distribution $\{a_k\}$ at arrival times, the probability vector $\mathbf{x}_k$ has to be rescaled by the appropriate $\mathbf{L}$ or $\widetilde{\mathbf{L}}$ matrix and then normalised

$$a_k = \begin{cases} G^{-1}(\mathbf{a}\mathbf{R}^k + \mathbf{b}\mathbf{S}^{B-1-k})\mathbf{L}\mathbf{e} & k = 0,\ldots,B-1 \\ G^{-1}\mathbf{x}_B\mathbf{T}^{k-B}\widetilde{\mathbf{L}}\mathbf{e} & k = B, B+1,\ldots \end{cases}.$$

G is a normalising constant given by

$$G = \mathbf{a}\frac{\mathbf{I} - \mathbf{R}^B}{\mathbf{I} - \mathbf{R}} + \mathbf{b}\frac{\mathbf{I} - \mathbf{S}^B}{\mathbf{I} - \mathbf{S}} + \mathbf{x}_B\frac{\mathbf{I}}{\mathbf{I} - \mathbf{T}}.$$

Other system properties such as the average queue length, the average waiting time etc. can be calculated from the steady state distributions $\{\mathbf{x}_k\}$ and $\{a_k\}$.

If the two MMPPs are modified N-Burst or N-Exp processes with average packet arrival rates $\lambda_{\text{avg}}^{(1)}$ and $\tilde{\lambda}_{\text{avg}}^{(1)}$ per source respectively, then the average packet arrival rate λ per source is

$$\lambda = \left(\sum_{k=0}^{B-1} \mathbf{x}_k\mathbf{e}\right)\lambda_{\text{avg}}^{(1)} + \left(1 - \sum_{k=0}^{B-1} \mathbf{x}_k\mathbf{e}\right)\tilde{\lambda}_{\text{avg}}^{(1)}. \tag{14}$$

The average arrival rate per source is thus the sum of the average arrival rates per source of MMPP numbers one and two, weighted by the relative time that the system spends in each MMPP.

Chapter 4

TCP Models: a Single Traffic Source

The TCP models discussed in this chapter assume a single persistent TCP source, *i.e.* a source which always has data to send, and calculate the throughput T as a function of some of the input parameters in table 4.1. Throughput is defined in definition 1. Only an overview of the models is given as well as the necessary equations for calculating the packet throughput per TCP connection.

e	the packet loss probability
λ_{TO}	the rate of TO losses occurring
λ_{TD}	the rate of TD losses occurring
$\overline{RTT}$	the average round trip time
T_0	the initial retransmission timeout value
w_{rx}	the maximum window size

Table 4.1: Input parameters to the models in chapter 4.

Definition 1 *The throughput T of a TCP connection is defined as the number of data packets successfully transmitted per second by the connection (the corresponding ACK packets are successfully received by the sender).*

4.1 The TCP-Ott Model

The TCP-Ott model (see Ott *et al.* [60]) calculates the throughput T as a function of $\overline{RTT}$ and e for low loss probabilities. Only CA and TD losses are modelled and it is assumed that multiple packet losses do not lead to timeouts. Ott *et al.* reasoned that the latter assumption is valid when modelling for example TCP-SACK and TCP-FACK.

TCP-Ott uses a fluid flow continuous time approximation to the discreet time process (W_n), where W_n is the congestion window after the successful acknowledgment of the n^{th} data packet that was sent. The mathematical derivation is lengthy and the reader is referred to [60] for more detail. The average window size $\overline{W}$ is derived as

$$\overline{W} = \frac{1.526912}{\sqrt{e}} \sim 1/\sqrt{e}. \tag{15}$$

Equation (15) is referred to as the $1/\sqrt{e}$ law for the TCP congestion window size and Ott *et al.* [60] gave the first formal proof of this law. The expression for T is

$$T = (1-e)\overline{W}/\overline{RTT}.$$

4.2 The TCP-Floyd Model

The TCP-Floyd model is described in Floyd [19] and Floyd and Fall [23], and was further extended in Floyd *et al.* [24]. It is a simple model and calculates the throughput T as a function of $\overline{RTT}$ and e.

TCP-Floyd is a deterministic model, and its calculation of the average congestion window size is illustrated in figure 4.1. The congestion window size W is initialised to $w_{\text{rx}}/2$ and is increased by

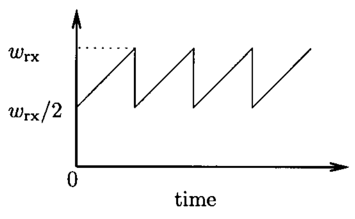


Figure 4.1: The evolution of the window size in the TCP-Floyd model.

1 every RTT. Each time that the window size reaches w_{rx}, a packet is dropped and the window size is halved. The average window size $\overline{W}$ is derived as

$$\overline{W} = \frac{-1.5e + \sqrt{6e + \frac{9}{4}e^2}}{2e}$$

which is approximated for small e by

$$\overline{W} = \frac{\sqrt{1.5}}{\sqrt{e}} \sim 1/\sqrt{e}. \tag{16}$$

Equation (16) also exhibits the $1/\sqrt{e}$ law. The throughput T is

$$T = (1-e)\overline{W}/\overline{RTT}.$$

4.3 The TCP-UMass1 Model

The TCP-UMass1 model (see Towsley *et al.* [62]) calculates the throughput T as a function of w_{rx}, e, $\overline{RTT}$ and T_0. It models CA, TO and TD losses but not SS. TCP-UMass1 is illustrated in figure 4.2.

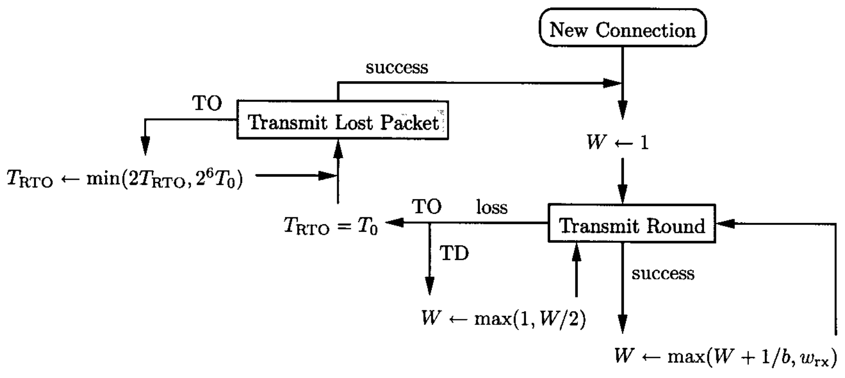

New Connection
success
TO
Transmit Lost Packet
$W \leftarrow 1$
$T_{\mathrm{RTO}} \leftarrow \min(2T_{\mathrm{RTO}}, 2^6 T_0)$
TO
loss
$T_{\mathrm{RTO}} = T_0$
Transmit Round
TD
success
$W \leftarrow \max(1, W/2)$
$W \leftarrow \max(W + 1/b, w_{\mathrm{rx}})$

Figure 4.2: The TCP-UMass1 model.

The TCP congestion avoidance behaviour is modelled in terms of *rounds*. A round starts with the back-to-back transmission of W packets where W is the current size of the TCP congestion window. W is initialized to one at the start of a connection. A round ends when the last of the W packets is acknowledged, indicating the start of the next round.

If all the packets in the round were sent successfully, the window size is increased by $1/b$ as long as $W < w_{\mathrm{rx}}$ where b is the number of data packets acknowledged by an ACK. Packet loss is indicated by either a TD or a TO. In the TD case, the window size is halved and packet transmission continues as before with the start of a new round. A TO occurs if no ACKs have been received for $T_{\mathrm{RTO}} = T_0$ time units. The lost packet is retransmitted after which the sender waits for $T_{\mathrm{RTO}} = 2T_0$ time units. If an ACK is received in that time, W is set to one and transmission is continued with the start of a new round. If no ACK has been received after T_{RTO} time units, T_{RTO} is set to $2T_{\mathrm{RTO}} = 4T_0$ and the lost packet is retransmitted. Each time that a TO occurs the lost packet is retransmitted and T_{RTO} is doubled until $T_{\mathrm{RTO}} = 2^6 T_0$ and remains constant thereafter until the packet is successfully transmitted whereupon $T_{\mathrm{RTO}} = T_0$.

The TCP-UMass1 model makes the following assumptions:

1. The duration of a round is equal to the RTT which is independent of the window size W at that time. (This implies that $\overline{RTT}$ is the average duration of a round.) This assumption was shown statistically to be valid for users with fast access links, but invalid for users at the end of a modem connection.

2. The time required to send all the packets in a round is smaller than the RTT.

3. If a packet is lost, the rest of the packets in the round are also lost. (Thus e is the probability that a packet is lost given that it is the first packet lost in its round.) This assumption tries to account for the drop-tail mechanism in many routers, where packets arriving to a full buffer are discarded resulting in multiple packet losses.

4. Packet loss in one round is independent of losses in other rounds. The reason for this is that packets in one round are separated from packets in other rounds by one RTT or more, and therefore they are likely to encounter buffer states that are independent of each other.

The TCP-UMass1 model yields the following equations

$$W(e) = \frac{2+b}{3b} + \sqrt{\frac{8(1-e)}{3be} + \left(\frac{2+b}{3b}\right)^2} \tag{17}$$

$$Q(e,w) = \min\left(1, \frac{(1-(1-e)^3)(1+(1-e)^3(1-(1-e)^{w-3}))}{1-(1-e)^w}\right) \tag{18}$$

$$f(e) = 1 + e + 2e^2 + 4e^3 + 8e^4 + 16e^5 + 32e^6. \tag{19}$$

$W(e)$ is the average window size of the TCP connection and $Q(e,w)$ is the probability that when a loss occurred in a round, it was due to a TO given that the window size was equal to w in that round. $f(e)$ is used in the following expression to calculate the average duration Z^{TO} of a timeout sequence

$$Z^{\text{TO}} = T_0 \frac{f(e)}{1-e}.$$

For $b = 1$ equation (17) for small values of e becomes

$$W(e) = 1 + \sqrt{\frac{8}{3}\frac{1}{e} - \frac{5}{3}} \approx \sqrt{\frac{8}{3}\frac{1}{e}} \sim 1/\sqrt{e}$$

which yields the $1/\sqrt{e}$ law. The throughput T is given by

$$T = (1-e)G$$

where

$$G = \begin{cases} \dfrac{\frac{1-e}{e} + W(e) + Q(e,W(e))\frac{1}{1-e}}{RTT(\frac{b}{2}W(e)+1) + Q(e,W(e))Z^{\text{TO}}} & W(e) < w_{\text{rx}} \\[2ex] \dfrac{\frac{1-e}{e} + w_{\text{rx}} + Q(e,w_{\text{rx}})\frac{1}{1-e}}{RTT(\frac{b}{8}w_{\text{rx}} + \frac{1-e}{ew_{\text{rx}}} + 2) + Q(e,w_{\text{rx}})Z^{\text{TO}}} & \text{otherwise} \end{cases}$$

which is approximated by

$$G = \min\left(\frac{w_{\text{rx}}}{RTT}, \frac{1}{RTT\sqrt{\frac{2be}{3}} + T_0 \min\left(1, 3\sqrt{\frac{3be}{8}}\right) e(1+32e^2)}\right).$$

4.4 The TCP-UMass2 Model

The TCP-UMass2 model (see Towsley *et al.* [61]) is an extension of the TCP-UMass1 model and calculates the throughput T using a Markov chain. It models TCP in the same way as the TCP-UMass1 model as explained in section 4.3. T is approximated as

$$T = \begin{cases} \dfrac{\frac{1-e}{e} + \frac{W(e)}{2} + Q(e, W(e))}{RTT(\frac{b}{2}W(e) + 1) + Q(e, W(e))Z^{\text{TO}}} & W(e) < w_{\text{rx}} \\[2ex] \dfrac{\frac{1-e}{e} + \frac{w_{\text{rx}}}{2} + Q(e, w_{\text{rx}})}{RTT\left(\frac{b}{8}w_{\text{rx}} + \frac{1-e}{ew_{\text{rx}}} + 2\right) + Q(e, w_{\text{rx}})Z^{\text{TO}}} & \text{otherwise} \end{cases} \tag{20}$$

where $W(e)$, $Q(e,x)$ and $f(e)$ are defined in the TCP-UMass1 model of section 4.3 in equations (17) to (19).

4.5 The TCP-UWash Model

The TCP-UWash model (see Cardwell *et al.* [12]) extends the results of the TCP-UMass1 and TCP-UMass2 models. It incorporates the impact of SS as well as the three way handshake for connection establishment, as these two mechanisms may dominate the transmission of small files. It calculates T as a function of the same parameters as the TCP-UMass1 model as well as the average length d (packets) of a file.

Apart from making the same assumptions as the TCP-UMass1 model, TCP-UWash also allows for small transfers which may or may not suffer packet loss. Recall from section 4.3 the expressions for $f(e)$, Z^{TO} and $W(e)$. The probability $Q(e,w)$ that a loss in CA is detected by a timeout is

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

The transmission of a file is assumed to proceed as illustrated in figure 4.3.

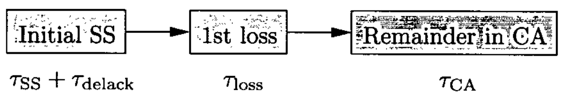


Figure 4.3: The process of transmission a file according to the TCP-UWash model.

d packets are transmitted using SS until all packets are transmitted or until the first loss occurs. After the first loss the rest of the packets are transmitted using CA. The expected time τ for a transfer is calculated as the sum of the four delays caused by the initial SS phase (denote τ_{SS}), the first packet loss (denote τ_{loss}), transmission of the remaining packets during CA (denote τ_{CA}) and a delayed acknowledgment (denote τ_{delack}) during the initial SS phase. Thus

$$\tau = \tau_{\text{SS}} + \tau_{\text{loss}} + \tau_{\text{CA}} + \tau_{\text{delack}}.$$

4.5.1 Initial SS

The average number of packets d_{SS} sent in the initial SS phase is

$$d_{\mathrm{SS}} = \begin{cases} d & e = 0 \\ 1 + \frac{(1-(1-e)^d)(1-e)}{e} & e > 0 \end{cases} .$$

The window size W_{SS} after sending d packets in SS without the slow start threshold constraint is

$$W_{\mathrm{SS}} = \frac{d_{\mathrm{SS}}(\gamma - 1)}{\gamma} + \frac{w_1}{\gamma}$$

where γ is the rate of growth of the congestion window during SS and w_1 is the size of the congestion window at the start of SS. Typically $\gamma = 2$ when one ACK is returned for every data packet and $1 \leq w_1 \leq 3$. τ_{SS} is then derived as

$$\tau_{\mathrm{SS}} = \begin{cases} \overline{RTT}\left(\log_\gamma (w_{\mathrm{rx}}/w_1) + 1 + \left(d_{\mathrm{SS}} - \frac{\gamma w_{\mathrm{rx}} - w_1}{\gamma - 1}\right)/w_{\mathrm{rx}}\right) & W_{\mathrm{SS}} > w_{\mathrm{rx}} \\ \\ \overline{RTT} \cdot \log_\gamma (d_{\mathrm{SS}}(\gamma - 1)/w_1 + 1) & W_{\mathrm{SS}} \leq w_{\mathrm{rx}} \end{cases} .$$

4.5.2 The First Packet Loss

The probability ℓ_{SS} that one or more packets are lost in SS due to timeout is

$$\ell_{\mathrm{SS}} = 1 - (1 - e)^d$$

and τ_{loss} is then derived as

$$\tau_{\mathrm{loss}} = \ell_{\mathrm{SS}} \left(Q(e, W_{\mathrm{SS}}) \left(Z^{\mathrm{TO}} - \overline{RTT}\right) + \overline{RTT}\right).$$

4.5.3 Transferring the Remainder

The number of packets d_{CA} outstanding after SS was exited due to a packet loss is

$$d_{\mathrm{CA}} = d - d_{\mathrm{ss}}.$$

τ_{CA} is then derived as

$$\tau_{\mathrm{CA}} = d_{\mathrm{CA}}/T'$$

where T' is the throughput as defined in equation (20) for the TCP-UMass2 model.

4.5.4 Delayed Acknowledgments

The expected delay τ_{delack} of the first delayed ACK is 100 milliseconds for BSD derived TCPs and 150 milliseconds for MS Windows.

The throughput T is given by $T = d/\tau$.

4.6 The TCP-UMass3 Model

The previous models analysed TCP's congestion avoidance behaviour from a source-centric point of view: the source injects packets into the network where they are lost with probability e. In [52], Towsley *et al.* take a different approach with the TCP-UMass3 model which models losses from a network centric point of view: the network is assumed to be a source of congestion and loss indications *arrive* to the TCP connection according to a Poisson process. CA and TD losses as well as single TO losses are modelled, but not multiple consecutive TO losses. TCP-UMass3 computes the throughput T as a function of w_{rx}, $\overline{RTT}$, T_0 and the rates λ_{TD} and λ_{TO}. It is a fluid flow model and is illustrated in figure 4.4.

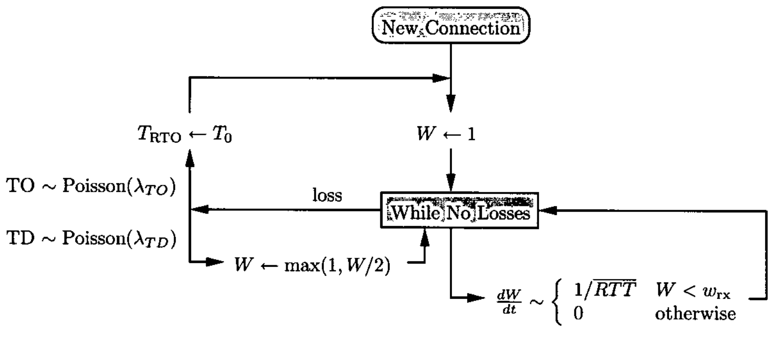

New Connection
$T_{\mathrm{RTO}} \leftarrow T_0$
$W \leftarrow 1$
TO ~ Poisson(λ_{TO})
TD ~ Poisson(λ_{TD})
loss
While No Losses
$W \leftarrow \max(1, W/2)$
$\frac{dW}{dt} \sim \begin{cases} 1/\overline{RTT} & W < w_{\mathrm{rx}} \\ 0 & \text{otherwise} \end{cases}$

Figure 4.4: The TCP-UMass3 Model

The increase in the window size W is continuous at a constant rate $1/\overline{RTT}$, which increases W by 1 every successful RTT. A round is successful if all the packets in that round is successfully transmitted and acknowledged. This increase continues until a loss occurs or until W equals the maximum window size w_{rx}. TD and TO losses are modelled as individual Poisson streams with rates λ_{TD} and λ_{TO} respectively. When a TD loss occurs, W is halved. When a TO occurs, W is set to 1 after which the connection waits for $T_{\mathrm{RTO}} = T_0$ time units before continuing transmission.

Let $K = 1/\overline{RTT}$. The TCP-UMass3 model yields the following equations

$$\begin{aligned}
\overline{W} &= \frac{(1-P_2)K + \lambda_{TO}}{\lambda_{TD}/2 + \lambda_{TO}} \\
P_1 &= \frac{2\lambda_{TO}^2 + 2\lambda_{TO} + \lambda_{TO}\lambda_{TD} + 2\lambda_{TO}K + 2K^2 + 2K}{(K+1)(2w_{\mathrm{rx}}\lambda_{TO} + w_{\mathrm{rx}}\lambda_{TD} + 2K)} \\
P_2 &= P_1 + \frac{\lambda_{TO}K(2\lambda_{TO} + \lambda_{TD})(w_{\mathrm{rx}} - 1)T_0}{(K+1)(2K + 2w_{\mathrm{rx}}\lambda_{TO} + w_{\mathrm{rx}}\lambda_{TD})}
\end{aligned} \tag{21}$$

where $\overline{W}$ is the average window size and P_1 is the probability that W is equal to w_{rx}. The second term on the right hand side of equation (21) is the probability that the growth of the window size is zero. The growth will be zero during periods when TCP retransmits a packet which has been lost due to a TO indication after which TCP has to wait for the corresponding ACK to return.

The throughput T is given by

$$T = (1 - e)(\overline{W} - \lambda_{TO} T_0)/\overline{RTT}.$$

4.7 The TCP-Hungary Model

The TCP-Hungary model (see Fekete *et al.* [18]) improves on the $1/\sqrt{e}$ law for packet loss probabilities in the range [0.001 : 0.05]. It models CA and TD losses only and calculates the throughput T as a function of $\overline{RTT}$ and e.

TCP-Hungary is a fluid model, similar to the TCP-UMass3 model, and uses mean field approximations to derive equations for the average window size $\overline{W}$. The derivation is lengthy and the reader is referred to the paper [18] for further detail.

The assumption is made that packets are lost independently of one another. $\overline{W}$ is derived for two scenarios: a LAN scenario where the link delay is small and delay is caused mostly by buffering, and a WAN scenario where the delay caused by buffering is small compared to the link delay

$$\overline{W} = \begin{cases} 1.5269/\sqrt{e} & \text{LAN} \\ \sqrt{c^2 + 2/e} - c & \text{WAN} \end{cases}.$$

c is a parameter used to model the length of time that a TCP connection has to wait before it can proceed with transmission following a reduction in the window size caused by a packet loss. It was numerically calculated as $c = 2.6285$.

The throughput T is given by

$$T = (1 - e)\overline{W}/\overline{RTT}.$$

4.8 The TCP-Uppsala Model

The TCP-Uppsala model (see Olsén and Kaj [57, 58, 59]) calculates the throughput T of a TCP Tahoe connection as a function of e, $\overline{RTT}$ and T_0. It only models SS and TO losses.

TCP-Uppsala is similar to the TCP-UMass1 type models. It also models transmission in terms of rounds and assumes constant round trip times and independent packet losses.

Define a cycle to be the interval between congestion window reductions. TCP-Uppsala derives the following equations

$$\begin{aligned}
A &= (1-e)/e \\
B &= (1-e)^2/e \\
Y &= \frac{\exp\left(\frac{e}{\sqrt{2}\ln 2}\right)\left(\mathrm{Ei}\left(\frac{e}{\sqrt{2}\ln 2}\right) - \mathrm{Ei}\left(\frac{ew_{\mathrm{rx}}}{\ln 2}\right)\right)}{\ln 2} + \frac{\exp\left(-e(w_{\mathrm{rx}} - 1/\sqrt{2})/\ln 2\right)}{ew_{\mathrm{rx}}} \\
Q(e) &= \frac{e(3 - 9e + 13e^2 - 9e^3 + 3e^4)}{(1-e+e^2)^3}
\end{aligned}$$

where A is the average number of packets transmitted in a cycle before the first loss, B is the average number of packets transmitted in a cycle after the first loss, Y is the average number of rounds needed to transmit A packets and $Q(e)$ is the probability that a loss is detected with a TO rather than a TD. The function $\mathrm{Ei}(k)$ is the Exponential integral defined as

$$\mathrm{Ei}(k) = \int_k^\infty \frac{\exp(-t)}{t} dt.$$

The throughput T is given by

$$T = (1-e)\frac{A + B - (1-e)^{w_{\mathrm{rx}}+1}}{Y + 1 + \left(\frac{T_0}{1-2e} - 1\right) Q(e)}.$$

Chapter 5

TCP Models: a Limited Number of Traffic Sources

This chapter investigates two analytic TCP models, the TCP-Engset and TCP-NBurst models, for the scenario where a fixed number N of ON/OFF sources offers TCP traffic to a network link. Both models implicitly take the behaviour of TCP into account by means of performance reduction factors which are explained below. Section 5.1 discusses the network setup which is followed by a description of the two models.

5.1 The Network Setup

The network setup is illustrated in figure 5.1.

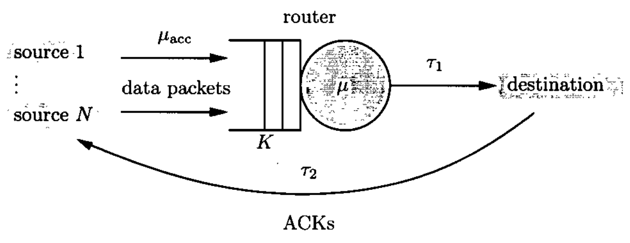


Figure 5.1: N TCP sources transmit packets over access links to a destination via a router.

N TCP sources transmit data packets over access links with capacity μ_{acc} (bits/s) to a router where the packets are queued in a buffer of length K (packets) until they are served. Packets are transmitted to some destination by the router which serves packets at a rate μ (bits/s). The destination responds to successfully received data packets by returning ACK packets to the appropriate sources. The latency between the router and the destination and between the destination and the sources are denoted by τ_1 and τ_2 (s) respectively.

A source transmits one file of average length φ (bits) at a time after which it goes into an OFF state which has an average duration of Z (seconds). While the source is transmitting, it is said to be in the ON state. Data and ACK packets are on average p_{usr} and p_{ack} (bits) long respectively. The maximum throughput rate $T_{\max}$ that can be achieved per file transmission is

$$T_{\max} = \mu_{\text{acc}}/p_{\text{usr}}.$$

(Throughput is defined in definition 1 in chapter 4.)

5.2 The TCP-Engset Model

The TCP-Engset model (see Heyman *et al.* [30]) is an adaptation of the Engset model (see appendix A.2) to describe the effects of TCP's flow control.

TCP-Engset was designed to model bursty webpage transfers by multiple users for the network setup in figure 5.1 where the capacities of the access links are less than the capacity of the router. The model calculates the throughput T per source as a function of the parameters in table 5.1.

N	the number of ON/OFF sources
Z	the average OFF time
μ_{acc}	the capacity (bits/s) of the access links
φ	the average file length (bits)
p_{usr}	the average packet length (bits)
μ	the capacity of the router (bits/s)
K	the buffer size (packets)

Table 5.1: Input parameters to the TCP-Engset model.

TCP-Engset is a flow-level model, which means that it views each active TCP connection as a constant bitstream, and not as a packet stream as for example the TCP-UMass1 model. CA and TD losses are included in the model by a performance reduction factor α which reduces the efficiency of the router during times of congestion.

A random number $J(t)$ of the N homogeneous ON/OFF sources are active (in the ON state) at time t. Let $L = \lfloor \mu/\mu_{\text{acc}} \rfloor$ denote the maximum number of sources that the router can handle without it being a bottleneck. When $j > L$ sources are active, the efficiency α of the router drops from 1 to $\alpha < 1$, at which time the aggregate service rate is $\alpha\mu$. This is illustrated in figure 5.2.

The following assumptions were made to simplify the model

1. The files that are transmitted are large enough so that CA is the dominating algorithm.

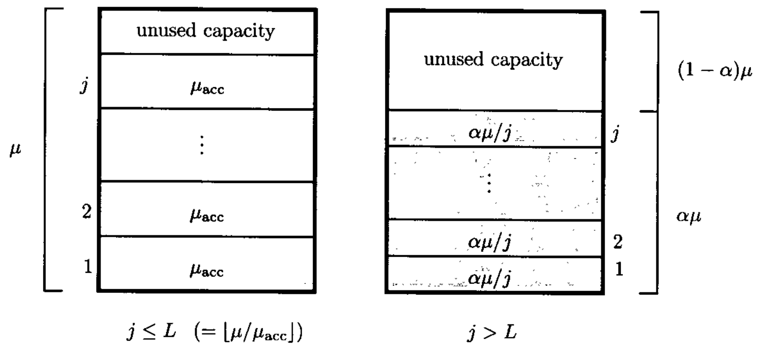


Figure 5.2: The bandwidth allocation per source when there is no congestion present in the network (left) and when congestion is present in the network (right).

2. The advertised window size is large enough so that the network is always the bottleneck.
3. The ON/OFF sources measure the same round trip times.
4. The j active sources are equally affected when congestion occurs at which time each source receives service at a rate $\alpha\mu/j$ (bits/s).

Let R_0 denote the RTT measured by sources when packets arrive to an empty queue

$$R_0 = p_{\text{usr}}/\mu + \tau_1 + \tau_2.$$

The estimation of α is based on how the congestion window evolves in TCP Reno. The derivation is presented in [30]. The expression for α is

$$\alpha = 1 - \frac{(\max(0, 1 - p_{\text{low}}))^2}{2(p_{\text{high}} - p_{\text{low}})} \tag{22}$$

where

$$p_{\text{high}} = \frac{K}{\mu R_0} + 1 \quad \text{and} \quad p_{\text{low}} = \frac{p_{\text{high}}}{2^{1.5}}.$$

p_{high} represents the number of packets in transit when the first overflow occurs. p_{low} represents the number of packets in transit when TCP proceeds with the CA algorithm after an overflow has concluded.

In [30, the appendix] it is shown that the stationary distribution $\{\pi_j\}$ of $J(t)$ is insensitive to the distributions of the file sizes and the OFF state intervals. $J(t)$ can thus be viewed as a birth-and-death process with birth and death rates

$$\begin{aligned} \beta_j &= \frac{N-j}{Z}, \quad j = 0, 1, \ldots, N \\ \delta_j &= \begin{cases} j\mu_{\text{acc}}/\varphi & j = 0, 1, \ldots, L \\ \alpha\mu/\varphi & j > L \end{cases}. \end{aligned} \tag{23}$$

Equation (23) shows that when $j > L$, the congestion in the router will cause the file transmission times to be extended. The stationary distribution $\{\pi_j\}$ can be calculated from the formula for a birth-and-death process

$$\pi_j = \pi_0 \frac{\beta_0 \dots \beta_{j-1}}{\delta_1 \dots \delta_j}, \quad j = 1, \dots, N$$

where π_0 is a normalising constant. Let $r(j)$ denote the transmission rate (packets/s) per TCP source when j sources are active

$$r(j) = \begin{cases} \frac{\mu_{\text{acc}}}{p_{\text{usr}}} & j \leq L \\ \frac{\alpha\mu}{j p_{\text{usr}}} & j > L \end{cases}. \tag{24}$$

The average throughput T per file transfer is given by

$$T = \sum_{j=1}^{N} r(j)\pi_j.$$

Finally, it should be noted that a solution exists for the steady state number of active sources $\{\boldsymbol{\pi}_j\}$ in the case of heterogeneous sources. Assume there are L source classes with N_ℓ sources in class ℓ. Further assume that the file sizes and OFF periods for class ℓ sources have distributions $F_\ell(\cdot)$ and $G_\ell(\cdot)$ respectively with means φ_ℓ and Z_ℓ respectively. Let $\gamma_\ell = \varphi_\ell / Z_\ell$ and let the vector $\boldsymbol{\pi}_j$ denote the number of active sources per source class at time t. The stationary distribution $\{\boldsymbol{\pi}_j\}$ is then given by

$$\boldsymbol{\pi}_j = \boldsymbol{\pi}_0 \frac{\prod_{\ell=1}^{L} \binom{N_\ell}{j_\ell} \gamma_\ell^{j_\ell}}{S(\sum_{\ell=1}^{L} j_\ell)}$$

where

$$S(n) = \prod_{m=1}^{n} r(m).$$

$\boldsymbol{\pi}_0$ is the appropriate normalising constant and $r(m)$ is defined in equation (24).

5.3 The TCP-*N*Burst Model

The TCP-*N*Burst model (see Schwefel [73]) uses a QBD process to model the network setup in section 5.1. It does not model TCP explicitly, but implicitly accounts for the burstiness of TCP traffic as well as its behaviour when congestion is detected in the network. It calculates various performance parameters such as the average throughput T per TCP source as a function of the parameters in tables 5.1 and 5.2.

B	the number of packets in the queue when congestion is detected
α	the throttling factor of the TCP-Engset model defined in equation (22)

Table 5.2: Additional input parameters for the TCP-*N*Burst model.

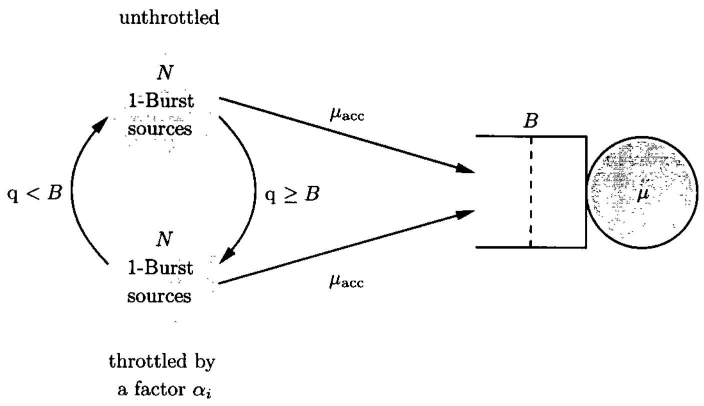


Figure 5.3: The TCP-NBurst model

The TCP-NBurst model is illustrated in figure 5.3.

The router has infinite buffer space and packet service times are exponentially distributed with parameter μ. A source transmits packets to the router over an access link with capacity μ_{acc} (bits/s). Traffic from the N sources is modelled as a modified N-Burst process (see section 3.8) for which the throttling factor α_i depends on the number of active ON/OFF sources. Let $\lambda_p = \mu_{\text{acc}}/p_{\text{usr}}$ denote the maximum transmission rate (packets/s) of a source. The average ON time U is

$$U = \frac{\varphi}{\mu_{\text{acc}}}.$$

Whenever the number of packets in the buffer is less than B, the arrival process is N-Burst with no throttling

$$\alpha_i = 1, \quad i = 1, \dots, N.$$

The assumption is made that when the buffer occupancy exceeds $B-1$, the j active sources detect congestion and they are equally throttled

$$\alpha_i = \alpha \min\left(\lambda_p, \frac{\mu}{j\lambda_p}\right).$$

TCP-NBurst uses the SM/M/1 queue in section 3.12 for which the average throughput per source is given in equation (14) to model the alternation between the two modified N-Burst arrival processes.

Chapter 6

TCP Models: An Infinite Number of Traffic Sources

This chapter discusses two analytic TCP models, the TCP-AK1 and TCP-AK2 models, for the scenario where an infinite number of sources offer TCP traffic to a network link. TCP traffic that arrives to the network is modelled in terms of TCP connections arriving per second. Each connection transmits one file and then terminates. Section 6.1 introduces the network setup used by both TCP models. A description is given of the two TCP models, followed by the TCP-AK1ext and TCP-AK2ext models which extend the TCP-AK1 and TCP-AK2 models respectively.

6.1 The Network Setup

The network setup under consideration is illustrated in figure 6.1 where users and servers are directly connected to an MPLS backbone network via ELRs I and E.

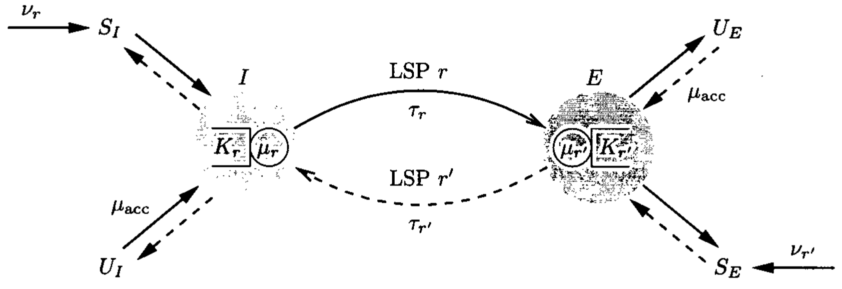


Figure 6.1: Users U_I, U_E and servers S_I, S_E are directly connected to an MPLS backbone network.

Routers I and E have buffer sizes K_r and $K_{r'}$ (packets) respectively and process packets at rates μ_r and $\mu_{r'}$ (bits/s) respectively. User U_E requests TCP connections from server S_I at rate ν_r (connections/s). During such a connection one file is transmitted to user U_E. Let φ denote the average length (bits) of a file. The data packets from server S_I travel over an access link to ELR I and on to ELR E via LSP r and finally to user U_E over an access link with capacity μ_{acc}. The bandwidth of the server access link is assumed to be infinite. The corresponding ACK packets travel from user U_E over the user access link to server S_I via ELR E, LSP r', ELR I and the server access link.

Connection requests to server S_E from user U_I are handled in a similar manner. Let p_{usr} and p_{ack} denote the average lengths (bits) of data and ACK packets respectively. LSPs r and r' have latencies τ_r and $\tau_{r'}$ (s) respectively. Finally, let e_r and $e_{r'}$ denote the loss probabilities at router I and E respectively. The maximum throughput rate $T_{\max}$ that can be achieved by a connection is

$$T_{\max} = \mu_{\text{acc}}/p_{\text{usr}}.$$

(Throughput is defined in definition 1 in chapter 4.)

6.2 The TCP-AK1 Model

The TCP-AK1 model (see Arvidsson and Krzesinski [6]) implicitly models CA and TD losses. It does not model SS or TO losses.

The model derives two fixed point equations to calculate the loads ρ_r and $\rho_{r'}$ on LSPs r and r' respectively. The model calculates many performance metrics including the throughput T per TCP connection. The input parameters to the TCP-AK1 model are summarised in table 6.1. In the analysis that follows, only the arguments and equations for calculating ρ_r are stated.

ν, ν'	the TCP connection request rates (connections/s)
φ	the average file size (bits)
w_{rx}	the maximum congestion window size (packets)
K_r, $K_{r'}$	the buffer sizes (packets)
p_{usr}	the average data packet size (bits)
p_{ack}	the average ACK packet size (bits)
μ_{acc}	the user access link capacity (bits/s)
μ_r, $\mu_{r'}$	the processing rates of the two routers I and E (bits/s)

Table 6.1: Input parameters to the TCP-AK1 model.

Figure 6.2 illustrates how TCP-AK1 models the network setup presented in section 6.1. Let γ_r and $\gamma_{r'}$ denote the average arrival rates (packets/s) of data packets from server S_I to LSP r and server S_E to LSP r' respectively. γ_r is thus the sum of the packet arrival rates of the individual TCP connections transmitting at server S_I.

Transmission over LSP r is modelled by an M/M/1/K queue where the arrival rate is the weighted sum of the data packet rate γ_r from S_I and the ACK packet rate $(1 - e_{r'})\gamma_{r'}$ from U_I.

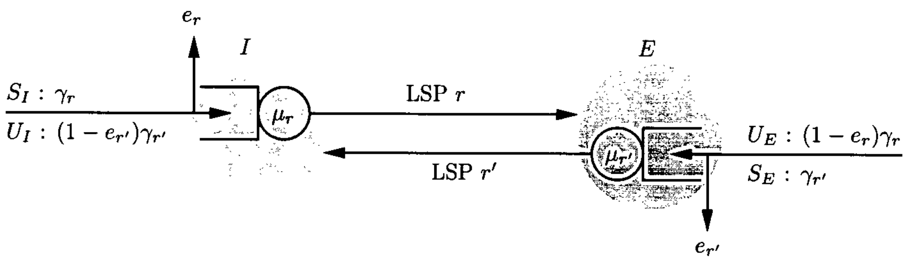


Figure 6.2: The TCP-AK1 model.

In order to simplify the model, the following assumptions are made

1. The data packets from a server arrive at the corresponding router according to a Poisson stream.
2. One ACK packet is returned for every data packet received. The ACK stream from user U_E therefore has the same arrival rate as the data packet output stream from router I.
3. The ACK packets from a user arrive at the corresponding router according to a Poisson stream.

Three steps are required to derive the fixed point equations that calculate the loads ρ_r and $\rho_{r'}$ on LSPs r and r' respectively. In the first step, the average RTT t_r measured by connections at S_I is calculated; step two computes the transmission rate λ_r of data packets per TCP connection; and step three derives the fixed point equations.

Step 1: This step assumes that we have a value for the load ρ_r which is used to calculate the round trip time t_r

$$t_r = q_r y_r + \frac{p_{\text{usr}}}{\mu_r} + \tau_r + \frac{p_{\text{usr}}}{\mu_{\text{acc}}} + \frac{p_{\text{ack}}}{\mu_{\text{acc}}} + q_{r'} y_{r'} + \frac{p_{\text{ack}}}{\mu_{r'}} + \tau_{r'}. \tag{25}$$

q_r and e_r are the average number of packets and the loss probability of the M/M/1/K queue given by

$$q_r = \begin{cases} \frac{\rho_r}{1-\rho_r} \frac{1-(K+1)\rho_r^K + K\rho_r^{K+1}}{1-\rho_r^{K+1}} & \rho_r \neq 1 \\ \frac{K}{2} & \rho_r = 1 \end{cases} \tag{26}$$

and

$$e_r = \begin{cases} \frac{1-\rho_r}{1-\rho_r^{K+1}} \rho_r^K & \rho_r \neq 1 \\ \frac{1}{K+1} & \rho_r = 1 \end{cases} \tag{27}$$

respectively. y_r is the average time to transmit a packet

$$y_r = \frac{\rho_r}{\gamma_r + (1 - e_{r'})\gamma_{r'}}. \tag{28}$$

The values for γ_r and $\gamma_{r'}$ are calculated from the equations

$$\rho_r = \frac{\gamma_r p_{\text{usr}} + (1 - e_{r'})\gamma_{r'} p_{\text{ack}}}{\mu_r} \tag{29}$$

$$\rho_{r'} = \frac{\gamma_{r'} p_{\text{usr}} + (1 - e_r)\gamma_r p_{\text{ack}}}{\mu_{r'}}. \tag{30}$$

Therefore

$$\gamma_r = \frac{p_{\text{usr}}\rho_r\mu_r - \rho_{r'}\mu_{r'}(1 - e_{r'})p_{\text{ack}}}{p_{\text{usr}}^2 - (1 - e_{r'})(1 - e_r)p_{\text{ack}}^2}$$
$$\gamma_{r'} = \frac{p_{\text{usr}}\rho_{r'}\mu_{r'} - \rho_r\mu_r(1 - e_r)p_{\text{ack}}}{p_{\text{usr}}^2 - (1 - e_{r'})(1 - e_r)p_{\text{ack}}^2}.$$

Step 2: The packet transmission rate λ_r per TCP connection on server S_I is

$$\lambda_r = \min\left(T_{\max}, w_r/t_r\right) \tag{31}$$

where w_r is the average window size given by the fixed point equation

$$w_{r+1} = (w_r + 1)(1 - e_r)^{w_r} + \frac{w_r}{2}\left(1 - (1 - e_r)^{w_r}\right). \tag{32}$$

Equation (32) models the CA algorithm. The first term models the increase of the congestion window by 1 every RTT and the second term models the halving of the congestion window when a TD loss occurs. If w_r is larger than the maximum window size w_{rx}, we set $w_r = w_{\text{rx}}$.

Equation (32) can be used to derive the $1/\sqrt{e}$ law for the window size w for small e where e is the packet loss probability. Thus

$$\begin{aligned} w &= (w + 1)(1 - e)^w + \frac{w}{2}\left(1 - (1 - e)^w\right) \\ &\approx (w + 1)(1 - ew) + \frac{w}{2}ew \qquad \text{for small } e. \end{aligned}$$

w is then given by the quadratic equation

$$ew^2 + 2ew - 2 = 0$$

which has the positive solution

$$w = \sqrt{1 + 2/e} - 1 \approx 1.414/\sqrt{e} \qquad \text{for small } e.$$

This derivation of the $1/\sqrt{e}$ law is simple compared to the derivation in Ott *et al.* [60] which uses a fluid-flow continuous time approximation to the discreet time process (W_n) defined in section 4.1.

Step 3: Let C_r denote the average number of TCP connections transmitting over LSP r

$$C_r = \alpha\nu_r\frac{n}{\lambda_r}. \tag{33}$$

α is an attraction factor [6] that models the increase in the TCP connection request rate when the throughput is high as well as the decrease in the connection request rate when throughput is low as a result of congestion in the network. n is the total number of packets sent (including retransmitted packets) in order to transmit a file of length φ

$$n = \frac{\varphi}{p_{\text{usr}}} \frac{1}{1 - e_r}.$$

An improved value for ρ_r can now be calculated by adjusting the load to match the connection data rates λ_r and $\lambda_{r'}$

$$\rho_r = \frac{C_r \lambda_r p_{\text{usr}} + C_{r'}(1 - e_{r'})\lambda_{r'} p_{\text{ack}}}{\mu_r}. \tag{34}$$

Similar versions of equations (25) through (34) for calculating the load on LSP r' are obtained by interchanging the indices r and r'. These equations define two simultaneous non-linear equations

$$\begin{aligned} \rho_r &= f_r(\rho_r, \rho_{r'}) \\ \rho_{r'} &= f_{r'}(\rho_{r'}, \rho_r) \end{aligned}$$

which are solved numerically by applying Newton's method with numerically computed Jacobians.

The throughput T_r per TCP connection on LSP r is

$$T_r = (1 - e_r)\lambda_r$$

where λ_r is calculated in equation (31).

6.3 The TCP-AK2 Model

The TCP-AK2 model (see Arvidsson and Krzesinski [5]) is similar to the TCP-AK1 model presented in section 6.2. It uses a system of two simultaneous fixed point equations to calculate the loads ρ_r and $\rho_{r'}$ on LSPs r and r' respectively. Each fixed point balances the interaction between two models: a TCP model and a network model. TCP-AK2 models SS, CA, TD losses and TO losses. In the description that follows, much detail has been left out. The reader is referred to the original text for more information.

6.3.1 The TCP Submodel

Assume that the window size is expressed in packets and that time is measured in units of the RTT τ. Assume further that file lengths are geometrically distributed with parameter ϕ. Thus $P(n \text{ packets}) = \phi^{n-1}(1 - \phi)$ with an average file length of $\varphi/(p_{\text{usr}} - p_{\text{ack}}) = 1/(1 - \phi)$ (packets).

Let q and s denote respectively the probabilities that a packet is received successfully and not received and acknowledged successfully

$$\begin{aligned} q &= 1 - e_r, \\ s &= 1 - (1 - e_r)(1 - e_{r'}). \end{aligned}$$

Finally, let w_{th} denote the slow start threshold *ssthresh*.

TCP-AK2 models the behaviour of TCP as an embedded Markov chain with five transient states SS′, SS, CA, DR and TR and one absorbing state OK. The transient states correspond respectively to TCP performing SS for the first time, SS but not for the first time, CA, a retransmission due to a TD loss and a retransmission due to a TO loss. TCP enters the OK state when it terminates successfully. While a TCP connection is transmitting a file, it transitions among some or all of these states before terminating successfully. Figure 6.3 illustrates the state transitions (the OK state is not included in the figure).

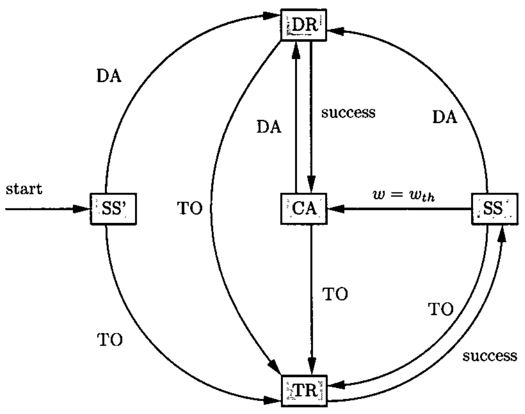


Figure 6.3: The transitions among the TCP states. The OK state is not included. Transitions to OK take place from SS′, SS, CA, DR and TR when file transmission is completed.

Transitions from SS′, SS and CA may be caused by the loss of a packet. In addition, a transition from SS to CA will occur when the threshold w_{th} is reached. A TD loss leads to state DR whereas a TO loss leads to state TR. If more data remain in DR, a transition to CA will occur if the packet retransmission is successful and to TR otherwise. Similarly, if data remain while in TR, a transition to SS will occur after successful packet retransmission.

A Single Visit to the SS′ State

The average number $N_{\text{SS}'}$ of packets transmitted while TCP is in the SS′ state (including the first unsuccessful packet) is

$$N_{\text{SS}'} = \frac{1}{1 - q^*} \tag{35}$$

where $q^* = \phi(1-e_r)$ is the probability that there is a packet and that it is successfully transmitted. The time $T_{\mathrm{SS}'}$ to transmit $N_{\mathrm{SS}'}$ packets is

$$T_{\mathrm{SS}'} = \begin{cases} \log_2(N_{\mathrm{SS}'}+1) & N_{\mathrm{SS}'} < w_{\mathrm{rx}} - 1 \\ \log_2 w_{\mathrm{rx}} - 1 + \frac{N_{\mathrm{SS}'}+1}{w_{\mathrm{rx}}} & N_{\mathrm{SS}'} \geq w_{\mathrm{rx}} - 1 \end{cases} . \tag{36}$$

The average window size $W_{\mathrm{SS}'}$ at which the first loss occurs is

$$W_{\mathrm{SS}'} = \min\left(N_{\mathrm{SS}'}\big|_{q^*=q}, w_{\mathrm{rx}} \right) . \tag{37}$$

In the last equation q^* is replaced by q as the average window size is only of interest if the file transmission has not yet completed. Finally, the probability $p_{\mathrm{TO}}(\mathrm{SS}')$ that a loss is detected by a TO is

$$p_{\mathrm{TO}}(\mathrm{SS}') = \min\left(1, \frac{3}{\phi N_{\mathrm{SS}'}}\right) . \tag{38}$$

Visits to the SS State

The equations are similar for subsequent visits to SS

$$N_{\mathrm{SS}} = \min(N_{\mathrm{SS}'}, w_{\mathrm{th}} - 1), \tag{39}$$

$$T_{\mathrm{SS}} = \begin{cases} \log_2(N_{\mathrm{SS}}+1) & N_{\mathrm{SS}} < w_{\mathrm{th}} - 1 \\ \log_2 w_{\mathrm{rx}} - 1 + \frac{N_{\mathrm{SS}}+1}{w_{\mathrm{rx}}} & N_{\mathrm{SS}} \geq w_{\mathrm{th}} - 1 \end{cases} , \tag{40}$$

$$W_{\mathrm{SS}} = \min(W_{\mathrm{SS}'}, w_{\mathrm{th}} - 1), \tag{41}$$

$$p_{\mathrm{TO}}(\mathrm{SS}) = \min\left(1, \frac{3}{\phi \min(N_{\mathrm{SS}}, w_{\mathrm{th}} - 1)}\right) . \tag{42}$$

Visits to the CA State

Similar to the previous equations, the average number N_{CA} of packets transmitted in the CA state (including the first unsuccessful packet) is

$$N_{\mathrm{CA}} = \frac{1}{1-q^*} \tag{43}$$

and the time T_{CA} to transmit N_{CA} packets is

$$T_{\mathrm{CA}} = \begin{cases} \frac{1-2w_{\mathrm{th}}}{2} + \sqrt{\left(\frac{1-2w_{\mathrm{th}}}{2}\right)^2 + 2N_{\mathrm{CA}}} & N_{\mathrm{CA}} < N \\ \frac{2N_{\mathrm{CA}} + (w_{\mathrm{rx}} - w_{\mathrm{th}})(w_{\mathrm{rx}} - w_{\mathrm{th}} + 1)}{2w_{\mathrm{rx}}} & N_{\mathrm{CA}} \geq N \end{cases} \tag{44}$$

where $N = (w_{\mathrm{rx}} - w_{\mathrm{th}})(w_{\mathrm{rx}} + w_{\mathrm{th}} - 1)/2$ is the number of packets sent in the time for the congestion window to reach w_{rx}. The average window size W_{CA} at which the first loss occurs during CA is

$$W_{\mathrm{CA}} = \min(w_{\mathrm{th}} + \widehat{T}_{\mathrm{CA}}, w_{\mathrm{rx}}) \tag{45}$$

where $\widehat{T}_{\mathrm{CA}}$ refers to Equation (44) with N_{CA} replaced by $(N_{\mathrm{CA}}-1)|_{q^*=q}$. The minus one accounts for the last packet being lost. The probability $p_{\mathrm{TO}}(\mathrm{CA})$ that a loss is detected by a TO is approximated as

$$p_{\mathrm{TO}}(\mathrm{CA}) = \min\left(1, \frac{3}{\phi(w_{\mathrm{th}} + \widehat{T}_{\mathrm{CA}})}\right). \tag{46}$$

State Transition Probabilities

The transition probability matrix of the embedded Markov chain is given by

$$\begin{pmatrix}
0 & 0 & 0 & \pi_{\mathrm{SS'},\mathrm{DR}} & \pi_{\mathrm{SS'},\mathrm{TR}} & \pi_{\mathrm{SS'},\mathrm{OK}} \\
0 & 0 & \pi_{\mathrm{SS},\mathrm{CA}} & \pi_{\mathrm{SS},\mathrm{DR}} & \pi_{\mathrm{SS},\mathrm{TR}} & \pi_{\mathrm{SS},\mathrm{OK}} \\
0 & 0 & 0 & \pi_{\mathrm{CA},\mathrm{DR}} & \pi_{\mathrm{CA},\mathrm{TR}} & \pi_{\mathrm{CA},\mathrm{OK}} \\
0 & 0 & \pi_{\mathrm{DR},\mathrm{CA}} & 0 & \pi_{\mathrm{DR},\mathrm{TR}} & \pi_{\mathrm{DR},\mathrm{OK}} \\
0 & \pi_{\mathrm{TR},\mathrm{SS}} & 0 & 0 & 0 & \pi_{\mathrm{TR},\mathrm{OK}} \\
0 & 0 & 0 & 0 & 0 & 1
\end{pmatrix}$$

where the π refer to the state transition probabilities. The elements in each row sum to one and all of the states are transient except for OK which is an absorbing state. The state transition probabilities π are

$$\begin{aligned}
\pi_{\mathrm{SS'},\mathrm{OK}} &= \sum_{n=1}^{\infty} \phi^{n-1}(1-\phi)q^n = q(1-\phi)\frac{1}{1-q\phi}, \\
\pi_{\mathrm{SS},\mathrm{OK}} &= \sum_{n=1}^{w_{\mathrm{th}}-1} \phi^{n-1}(1-\phi)q^n \\
&= q(1-\phi)\frac{1-(q\phi)^{w_{\mathrm{th}}-1}}{1-q\phi}, \\
\pi_{\mathrm{CA},\mathrm{OK}} &= \sum_{n=1}^{\infty} \phi^{n-1}(1-\phi)q^n = q(1-\phi)\frac{1}{1-q\phi}, \\
\pi_{\mathrm{DR},\mathrm{OK}} &= (1-\phi)(1-s), \quad \pi_{\mathrm{TR},\mathrm{OK}} = (1-\phi)(1-s), \\
\pi_{\mathrm{SS},\mathrm{CA}} &= \sum_{n=w_{\mathrm{th}}}^{\infty} \phi^{n-1}(1-\phi)\, q^{w_{\mathrm{th}}-1} = (q\phi)^{w_{\mathrm{th}}-1}, \\
\pi_{\mathrm{SS'},\mathrm{DR}} &= (1-\pi_{\mathrm{SS'},\mathrm{OK}})(1-p_{\mathrm{TO}}(\mathrm{SS'})), \\
\pi_{\mathrm{SS'},\mathrm{TR}} &= (1-\pi_{\mathrm{SS'},\mathrm{OK}})p_{\mathrm{TO}}(\mathrm{SS'}), \\
\pi_{\mathrm{SS},\mathrm{DR}} &= (1-\pi_{\mathrm{SS},\mathrm{OK}}-\pi_{\mathrm{SS},\mathrm{CA}})(1-p_{\mathrm{TO}}(\mathrm{SS})), \\
\pi_{\mathrm{SS},\mathrm{TR}} &= (1-\pi_{\mathrm{SS},\mathrm{OK}}-\pi_{\mathrm{SS},\mathrm{CA}})p_{\mathrm{TO}}(\mathrm{SS}), \\
\pi_{\mathrm{CA},\mathrm{DR}} &= (1-\pi_{\mathrm{CA},\mathrm{OK}})(1-p_{\mathrm{TO}}(\mathrm{CA})), \\
\pi_{\mathrm{CA},\mathrm{TR}} &= (1-\pi_{\mathrm{CA},\mathrm{OK}})p_{\mathrm{TO}}(\mathrm{CA}), \\
\pi_{\mathrm{DR},\mathrm{CA}} &= \phi(1-s), \quad \pi_{\mathrm{DR},\mathrm{TR}} = s, \quad \pi_{\mathrm{TR},\mathrm{SS}} = \phi.
\end{aligned} \tag{47}$$

The average number V_θ of visits to each state θ is

$$\begin{aligned}
V_{SS'} &= V_{OK} = 1,\\
V_{SS} &= V_{TR}\pi_{TR,SS},\\
V_{CA} &= V_{SS}\pi_{SS,CA} + V_{DR}\pi_{DR,CA},\\
V_{DR} &= V_{SS'}\pi_{SS',DR} + V_{SS}\pi_{SS,DR} + V_{CA}\pi_{CA,DR},\\
V_{TR} &= V_{SS'}\pi_{SS',TR} + V_{SS}\pi_{SS,TR} + V_{CA}\pi_{CA,TR} + V_{DR}\pi_{DR,TR}
\end{aligned}$$

which yield

$$\begin{aligned}
V_{TR} &= ((1-\pi_{DR,CA}\pi_{CA,DR})\pi_{SS',TR} + (\pi_{DR,CA}\pi_{CA,TR} + \pi_{DR,TR})\pi_{SS',DR}) / ((1-\pi_{DR,CA}\pi_{CA,DR})\\
&\quad (1-\pi_{TR,SS}\pi_{SS,TR} - \pi_{TR,SS}\pi_{SS,CA}\pi_{CA,TR}) - (\pi_{DR,CA}\pi_{CA,TR} + \pi_{DR,TR}) \times\\
&\quad (\pi_{TR,SS}\pi_{SS,DR} + \pi_{TR,SS}\pi_{SS,CA}\pi_{CA,DR}))\,,\\
V_{DR} &= (\pi_{SS',DR} + V_{TR}\ (\pi_{TR,SS}\pi_{SS,DR} + \pi_{TR,SS}\pi_{SS,CA}\pi_{CA,DR}))\,/\,(1-\pi_{DR,CA}\pi_{CA,DR})\,,\\
V_{CA} &= V_{TR}\pi_{TR,SS}\pi_{SS,CA} + V_{DR}\pi_{DR,CA},\\
V_{SS} &= V_{TR}\pi_{TR,SS}. \qquad (48)
\end{aligned}$$

The Slow Start Threshold

Whenever TCP leaves the SS and CA states as a result of packet loss, the slow start threshold is updated as

$$w_{th}^{SS} = \max(2, W_{SS}/2) \quad \text{and} \quad w_{th}^{CA} = \max(2, W_{CA}/2)$$

for SS and CA respectively. The number of losses g_{SS} and g_{CA} in each state is

$$\begin{aligned}
g_{SS} &= V_{SS'}(\pi_{SS',DR} + \pi_{SS',TR}) + V_{SS}(\pi_{SS,DR} + \pi_{SS,TR}),\\
g_{CA} &= V_{CA}(\pi_{CA,DR} + \pi_{CA,TR}).
\end{aligned}$$

A new estimate for the slow start threshold w_{th} is then calculated by weighting the two cases

$$w_{th} = \frac{g_{SS}}{g_{SS}+g_{CA}} w_{th}^{SS} + \frac{g_{CA}}{g_{SS}+g_{CA}} w_{th}^{CA}. \qquad (49)$$

Equations (35) to (49) define a fixed point equation

$$w_{th} = f(w_{th}) \qquad (50)$$

which is solved for w_{th}. Once a value for w_{th} has been calculated we set

$$w_{th} = \begin{cases} 2 & w_{th} < 2 \\ w_{rx} & w_{th} > w_{rx} \\ w_{th} & \text{otherwise} \end{cases}.$$

The lower limit is imposed by the protocol while the upper limit is the restriction placed on the congestion window by the receiver.

TD Retransmission

During DR only one packet is transmitted hence $N_{\mathrm{DR}} = 1$. The time T_{DR} spend in DR is approximated as

$$T_{\mathrm{DR}} = R/\tau \tag{51}$$

where R is the RTT (seconds) defined in equation (54) below. R is divided by τ so that T_{DR} is in units of the RTT.

TO Retransmission

The average time T_{TR} spent in TR is

$$T_{\mathrm{TR}} = \frac{E+4D}{\tau}\left(\frac{1-(2s)^{M-B}}{1-2s} + \frac{(2s)^{M-B}}{1-s}\right)2^B. \tag{52}$$

E and D are respectively the mean and an approximation of the mean deviation of the RTT (seconds) where each is the weighted sum of an initial value and a steady state expectation

$$\begin{aligned} E &= g_0 E_0 + g_\infty E_\infty, \\ D &= g_0 D_0 + g_\infty D_\infty. \end{aligned}$$

The expressions for E_0, E_∞, D_0 and D_∞ are given in the description of the network submodel in section 6.3.2. The weights g_0 and g_∞ are

$$\begin{aligned} g_0 &= \frac{1-\phi}{1-e_r}\left(\frac{\ln(1-\phi e_r) - \ln(1-\phi)}{\phi} - \frac{e_r(1-e_r)}{1-\phi e_r}\right), \\ g_\infty &= 1 - g_0. \end{aligned}$$

$B = \min\left((B_1'/B_1)B_{\mathrm{n}}B_1', M\right)$ is the average number of consecutive retransmissions seen by an arbitrary packet (the tagged packet). B_{n} is the average number of consecutive lost packets that precede the tagged packet

$$B_n = \frac{e_r}{1-e_r}\left(1 - \frac{1-\phi}{1-e_r}\frac{\ln(1-\phi e_r) - \ln(1-\phi)}{\phi}\right)$$

and B_1 is the average number of retransmissions per lost packet

$$B_1 = 1/(1-e_r).$$

B_1' is a correction term that accounts for the limit on the number of packets that can be retransmitted in order to send one data packet successfully

$$B_1' = \log_2\left(\frac{2(1-e_r) - (2e_r)^M}{1-2e_r}\right).$$

$M = 6$ is the number of times that the retransmission timeout value T_{RTO} is multiplied by two before it is kept constant at $2^6 T_0$ where T_0 is the initial retransmission timeout value at the start of consecutive timeouts. T_0 is initialised to 6 (seconds) at the start of the connection and has a steady state average value of $T_0 = E + 4D$.

6.3.2 The Network Submodel

TCP-AK2 models the network setup presented in section 6.1 in the same way as the TCP-AK1 model as shown in figure 6.2. Recall the network parameters in section 6.1 and from section 6.2 equation (28) the average time y_r to transmit a packet as well as the expressions for the M/M/1/K queue loss probability e_r (equation (27)) and average number of packets q_r (equation (26)) in the queue.

The Round Trip Time

The sums D_r and $D_{r'}$ of the average queuing and transmission times for a data packet traveling from S_I to U_E and for an ACK packet traveling from U_E to S_I are

$$D_r = q_r y_r + p_{\mathrm{usr}}/\mu_{\mathrm{acc}} \quad \text{and} \quad D_{r'} = p_{\mathrm{ack}}/\mu_{\mathrm{acc}} + q_{r'} y_{r'}. \tag{53}$$

Transmission is said to be saturated when the ACK for the first data packet in a window catches up with the last data packet to be sent in the same window, and unsaturated otherwise. The average round trip time R when transmission is unsaturated is

$$R = D_r + \tau_r + D_{r'} + \tau_{r'}. \tag{54}$$

The average RTT τ is derived by considering the periods when transmission is saturated and unsaturated. Let D_{max} denote the maximum of the queuing and transmission times on any link or LSP and at any router

$$D_{\mathrm{max}} = \max\left(q_r y_r, \frac{p_{\mathrm{usr}}}{\mu_{\mathrm{acc}}}, q_{r'} y_{r'}, \frac{p_{\mathrm{ack}}}{\mu_{\mathrm{acc}}}\right)$$

The average times $\{S_\theta | \theta \in \{\mathrm{SS}', \mathrm{SS}, \mathrm{CA}\}\}$ (expressed in units of the RTT) to reach saturation are respectively

$$S_{\mathrm{SS}'} = S_{\mathrm{SS}} = \log_2(R/D_{\mathrm{max}}) \quad \text{and} \quad S_{\mathrm{CA}} = (R/D_{\mathrm{max}}) - w_{\mathrm{th}}$$

and the times $\{U_\theta | \theta \in \{\mathrm{SS}', \mathrm{SS}, \mathrm{CA}\}\}$ (expressed in units of the RTT) to reach the window limit w_{rx} in the absence of packet loss are respectively

$$U_{\mathrm{SS}'} = U_{\mathrm{SS}} = \log_2 w_{\mathrm{rx}} \quad \text{and} \quad U_{\mathrm{CA}} = w_{\mathrm{rx}} - w_{\mathrm{th}}.$$

Let

$$\begin{aligned}
\Omega(\mathrm{S}_\theta) &= \{1, \ldots, \min(S_\theta, U_\theta, T_\theta)\}, \\
\Omega(\mathrm{U}_\theta) &= \{\min(S_\theta, U_\theta, T_\theta) + 1, \ldots, \min(U_\theta, T_\theta)\}, \\
\Omega(\mathrm{T}_\theta) &= \{\min(U_\theta, T_\theta) + 1, \ldots, T_\theta\}
\end{aligned}$$

be an enumeration and define

$$\begin{aligned}
|\Omega(\mathrm{S}_\theta)| &= \min(S_\theta, U_\theta, T_\theta), \\
|\Omega(\mathrm{U}_\theta)| &= \min(U_\theta, T_\theta) - \min(S_\theta, U_\theta, T_\theta), \\
|\Omega(\mathrm{T}_\theta)| &= T_\theta - \min(U_\theta, T_\theta).
\end{aligned}$$

Then the sums $\{\overline{\tau}_\theta | \theta \in \{\mathrm{SS}', \mathrm{SS}, \mathrm{CA}\}\}$ of the times spent in each state θ are

$$\begin{aligned}
\overline{\tau}_{\mathrm{SS}'} &= |\Omega(\mathrm{S}_{\mathrm{SS}'})|R + |\Omega(\mathrm{T}_{\mathrm{SS}'})| \max(R, w_{\mathrm{rx}} D_{\mathrm{max}}) + 2^{|\Omega(S_{\mathrm{SS}'})|}(2^{|\Omega(U_{\mathrm{SS}'})|} - 1) D_{\mathrm{max}},\\
\overline{\tau}_{\mathrm{SS}} &= |\Omega(\mathrm{S}_{\mathrm{SS}})|R + |\Omega(\mathrm{T}_{\mathrm{SS}})| \max(R, w_{\mathrm{rx}} D_{\mathrm{max}}) + 2^{|\Omega(S_{\mathrm{SS}})|}(2^{|\Omega(U_{\mathrm{SS}})|} - 1) D_{\mathrm{max}},\\
\overline{\tau}_{\mathrm{CA}} &= |\Omega(\mathrm{S}_{\mathrm{CA}})|R + |\Omega(\mathrm{T}_{\mathrm{CA}})| \max(R, w_{\mathrm{rx}} D_{\mathrm{max}}) + |\Omega(U_{\mathrm{CA}})|(2w_{\mathrm{th}} - 1 + |\Omega(U_{\mathrm{CA}})| + 2|\Omega(S_{\mathrm{CA}})|) D_{\mathrm{max}}/2.
\end{aligned}$$

The average RTT τ (s) is

$$\tau = \frac{\overline{\tau}_{\mathrm{SS}'} + V_{\mathrm{SS}}\overline{\tau}_{\mathrm{SS}} + V_{\mathrm{CA}}\overline{\tau}_{\mathrm{CA}}}{T_{\mathrm{SS}'} + V_{\mathrm{SS}}T_{\mathrm{SS}} + V_{\mathrm{CA}}T_{\mathrm{CA}}}. \tag{55}$$

The Average and Variance of the Round Trip Time

The initial values E_0 and D_0 for the average and variance of the round trip time respectively are assigned by the protocol as $E_0 = 0$ and $D_0 = 1.5$. The corresponding expected long term values for E_∞ and D_∞ are approximated as

$$E_\infty = \tau \quad \text{and } D_\infty = \sqrt{S_r^2 + S_{r'}^2 + \sigma_w^2}.$$

S_r^2 and $S_{r'}^2$ are the variances of the transmission times on LSP r and LSP r' respectively which are equal to the respective M/M/1/K waiting time variances (see appendix D.) The σ_w^2 component represents the variance of the round trip time caused by variations in the window size and is approximated by

$$\sigma_w = (t_{\mathrm{hi}} - t_{\mathrm{lo}})/2$$

where

$$\begin{aligned}
t_{\mathrm{hi}} = &(V_{\mathrm{SS}'}(1 - \pi_{\mathrm{SS}',\mathrm{OK}}) \min(N_{\mathrm{SS}'}, w_{\mathrm{rx}}) + \; V_{\mathrm{SS}}(1 - \pi_{\mathrm{SS},\mathrm{OK}}) \min(N_{\mathrm{SS}}, w_{\mathrm{th}}) +\\
&V_{\mathrm{CA}}(1 - \pi_{\mathrm{CA},\mathrm{OK}}) \min(w_{\mathrm{th}} + T_{\mathrm{CA}}, w_{\mathrm{rx}}))\, D_{\mathrm{max}}\\
&/(V_{\mathrm{SS}'}(1 - \pi_{\mathrm{SS}',\mathrm{OK}}) + V_{\mathrm{SS}}(1 - \pi_{\mathrm{SS},\mathrm{OK}}) + V_{\mathrm{CA}}(1 - \pi_{\mathrm{CA},\mathrm{OK}})),\\
t_{\mathrm{lo}} = &\frac{V_{\mathrm{TR}}(1 - \pi_{\mathrm{TR},\mathrm{OK}}) + V_{\mathrm{DR}}(1 - \pi_{\mathrm{DR},\mathrm{OK}}) w_{\mathrm{th}}}{V_{\mathrm{TR}}(1 - \pi_{\mathrm{TR},\mathrm{OK}}) + V_{\mathrm{DR}}(1 - \pi_{\mathrm{DR},\mathrm{OK}})} D_{\mathrm{max}}.
\end{aligned}$$

t_{hi} and t_{lo} are respectively the maximum and minimum values of the RTT.

The Packet Arrival Rate Per Connection

For an average file transmission N_{tot} packets are sent in T_{tot} time where

$$N_{\mathrm{tot}} = N_{\mathrm{SS}'} + V_{\mathrm{SS}}N_{\mathrm{SS}} + V_{\mathrm{CA}}N_{\mathrm{CA}} + V_{\mathrm{DR}}N_{\mathrm{DR}} + V_{\mathrm{TR}}N_{\mathrm{TR}}, \tag{56}$$

$$T_{\mathrm{tot}} = T_{\mathrm{SS}'} + V_{\mathrm{SS}}T_{\mathrm{SS}} + V_{\mathrm{CA}}T_{\mathrm{CA}} + V_{\mathrm{DR}}T_{\mathrm{DR}} + V_{\mathrm{TR}}T_{\mathrm{TR}}. \tag{57}$$

The data packet transmission rate λ_r (packets/s) per TCP connection is

$$\lambda_r = \min\left(T_{\mathrm{max}}, \frac{N_{\mathrm{tot}}}{\tau T_{\mathrm{tot}}}\right). \tag{58}$$

The Total Packet Arrival Rate

The average number C_r of TCP connections in progress on LSP r is $C_r = \nu_r N_{\text{tot}}/\lambda_r$ where ν_r is the TCP connection request rate on LSP r (see section 6.1). The load is adjusted to match the data packet transmission rates λ_r and $\lambda_{r'}$ of the TCP connections

$$\rho_r = \frac{C_r \lambda_r p_{\text{usr}} + C_{r'} \lambda_{r'} (1 - e_{r'}) p_{\text{ack}}}{\mu_r}. \tag{59}$$

Equations (35) through (59) define a system of two simultaneous non-linear equations

$$\rho_r = f_r(\rho_r, \rho_{r'}) \quad \text{and} \quad \rho_{r'} = f_{r'}(\rho_r, \rho_{r'})$$

which, as for the TCP-AK1 model, are solved numerically by applying Newton's method with numerically computed Jacobians. Figure 6.4 illustrates how the TCP and network submodels are

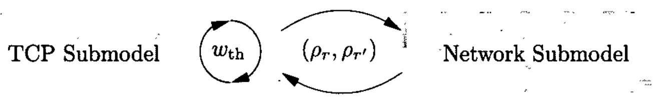


Figure 6.4: The TCP-AK2 model is solved by calculating the solutions to the TCP and network submodels repeatedly until the loads converge.

solved repeatedly until the dual fixed point is reached for the loads ρ_r and $\rho_{r'}$. Each time that the TCP submodel is solved, a fixed point equation for calculating the slow start threshold w_{th} has to be solved.

The throughput T_r per TCP connection on LSP r is

$$T_r = (1 - e_r)\lambda_r$$

where λ_r is calculated in equation (58).

6.4 Extending the TCP-AK1 and TCP-AK2 Models

In this section we extend the TCP-AK1 and TCP-AK2 models as the TCP-AK1ext and TCP-AK2ext models respectively. The first extension is the replacement of the M/M/1/K queue with the SM/M/1/K queue of section 3.11.

The SM/M/1/K queue was chosen in order to implement more complex arrival processes such as the N-Burst and N-Exp MMPPs. Let N-Burst/M/1/K and N-Exp/M/1/K denote the SM/M/1/K queue with the N-Burst and N-Exp arrival processes respectively. The SM/M/1/K queue has an analytic solution for metrics such as the average queue length and the loss probability, and is more tractable than for example the SM/D/1/K type queues (see Akar and Arikan [1] and Blondia and Geerts [8]) and the SM/G/1 and SM/D/1 type queues (see Lucantoni [42, 43]). (We used the notation SM loosely here and may include MAP (Markov Arrival Process), BMAP (Batch Markov Arrival Processes) and other arrival processes.)

The average queue length q and the packet loss probability e for the SM/M/1/K queue are given in equations (10) and (11) respectively. The variance of the waiting time in the queue can be calculated from the equations in appendix D.

The additional parameters to the TCP-AK1ext and TCP-AK2ext models are summarised in table 6.2.

m	the number of phases in the TPT distribution (for N-Burst only)
ε	the ratio $\lambda_{\text{low}} : \lambda_{\text{ON}}$

Table 6.2: Additional input parameters to the TCP-AK1ext and TCP-AK2ext models.

For the N-Exp/M/1/K queue, ε was set to zero and the average OFF time Z_r and ON time U_r of connections transmitting on LSP r are

$$Z_r = T_{\max} - \lambda_r$$
$$U_r = \lambda_r.$$

The values of Z_r and U_r make no difference to the results provided the ratio Z_r/U_r is constant. Multiplying Z_r and U_r by a constant factor x is equivalent to dividing the matrix $\mathbf{Q}_N$ in equation (7) by x which does not influence the steady state distribution of $\mathbf{Q}_N$.

6.4.1 Further Alterations to the TCP-AK1 model

1. As TCP may spend more time in SS than in CA when transmitting small files or when packet loss rates are high, equation (32) for the average window size is changed to the following fixed point equation

$$w_r = (\gamma w_r)(1 - e_r)^{w_r} + (1 - (1 - e_r)^{w_r}) \tag{60}$$

   where γ is the growth rate of the congestion window during SS. As with the TCP-UWash model of section 4.5 γ is assigned a value of 2 because in the simulations to follow one ACK is returned for every data packet received. The first term represents the near-exponential growth of the congestion window and the second term represents the setting of the congestion window to one after a loss indication.

2. Table 6.3 presents the evolution of the window size in SS in the absence of packet loss and without window limitations.

round no.	1	2	3	4	5	6	⋯	t
cwnd	1	2	4	8	16	32	⋯	2^{t-1}
packets sent	1	3	7	15	31	63	⋯	$2^t - 1$

Table 6.3: The evolution of the congestion window in SS in the absence of packet loss and window limitations.

The first row shows the round number and the second row shows the evolution of the congestion window for every round. The third row shows the total number of packets sent at the end of each round. Table 6.3 shows that if we transmit a file of length 31 (packets), the average congestion window is equal to 31/5. Therefore, given that we are transmitting files of average length $d = \varphi/(p_{\text{usr}} - p_{\text{ack}})$ (packets), the average window size should not exceed $\overline{W}_{\text{max}}$ where

$$\overline{W}_{\text{max}} = d/\log_2(d+1). \tag{61}$$

Therefore, after a value for w_r has been calculated with equation (60), we set

$$w_r = \max(1, \min(w_r, \overline{W}_{\text{max}}, w_{\text{rx}})).$$

3. Let R_{min} denote the minimum RTT

$$R_{\text{min}} = \frac{p_{\text{usr}}}{\mu_{\text{acc}}} + \frac{p_{\text{usr}}}{\mu_r} + \tau_r + \frac{p_{\text{ack}}}{\mu_{\text{acc}}} + \frac{p_{\text{usr}}}{\mu_{r'}} + \tau_{r'}. \tag{62}$$

Then $\overline{W}_{\text{max}}/R_{\text{min}}$ is another upper bound for the throughput. Therefore

$$T_{\text{max}} = \min\left(\frac{\overline{W}_{\text{max}}}{R_{\text{min}}}, \frac{\mu_{\text{acc}}}{p_{\text{usr}}}\right). \tag{63}$$

Equation (63) can have a significant effect on the results. For example, if we are transmitting files of average length $d = 30$ (packets) over user access links with capacity $\mu_{\text{acc}} = 2$ (Mbits/s) and with $p_{\text{usr}} = 1540 \times 8$ (bits) and $R_{\text{min}} = 0.2$ (seconds), then

$$T_{\text{max}} \approx \min(30, 162) = 30 \ll 162 \text{ (packets/s)}.$$

4. We furthermore re-derive λ_r in equation (31) as

$$\lambda_r = \frac{A}{B + (A/w)t_r} \tag{64}$$

where A is the total number of packets (including retransmissions) required to transmit the file

$$A = \frac{\varphi}{p_{\text{usr}} - p_{\text{ack}}} \frac{1}{1 - e_r} \tag{65}$$

and B is the average time required to perform connection establishment

$$B = t_r + \sum_{i=1}^{4} 2^{i-1} T_0 s^i (1-s) + \sum_{i=5}^{7} 64 s^i (1-s) \tag{66}$$

with T_0 initialised to 6 at the start of the connection. The retransmission time value can increase up to a maximum of 64 seconds (see section 2.6.2). s was defined in section 6.3.1 as the probability of a loss on LSP r and LSP r'. The summation was chosen to be over 7 terms as most TCP implementations will terminate connection establishment after 7 failed attempts to connect. The second term in the denominator of equation (64) is the time to transmit the file which is equal to the number of required RTTs (equal to A/w) times the RTT t_r.

6.4.2 A Further Alteration to the TCP-AK2 Model

We extend the TCP-AK2 model to model connection establishment by adding the average time B for connection establishment in equation (66) to equation (56) for the total time T_{tot} to download a file

$$T_{\text{tot}} = B + T_{\text{SS}'} + V_{\text{SS}}T_{\text{SS}} + V_{\text{CA}}T_{\text{CA}} + V_{\text{DR}}T_{\text{DR}} + V_{\text{TR}}T_{\text{TR}}.$$

In addition, the new limit T_{max} on the throughput in equation (63) is imposed.

Chapter 7

Simulation and Results

This chapter compares the performance predictions of the various TCP models against the results obtained from simulation experiments. First the simulator and the simulation are discussed, followed by remarks on the applicability of the TCP models and information on how these models were used. Finally, some results are given.

7.1 Simulation

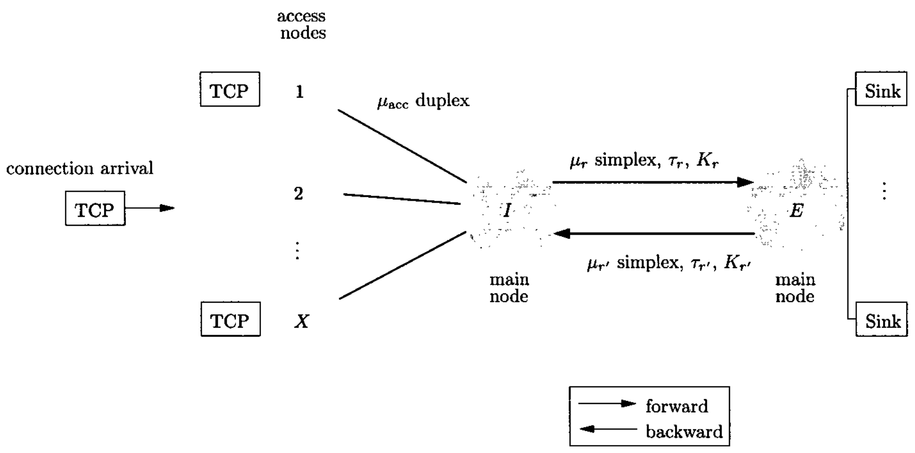
ns [1] Version 2 (2.1b8a) was used for the simulation experiments and TCP Reno was simulated with one ACK returned for every data packet received. *ns* is a packet level simulator, and its representation of the TCP Reno protocol is said to be sufficiently accurate for research purposes (see MCanne and Floyd [49]). To utilise *ns*, one needs to write a Tcl script as input to the simulator. Section 7.1.1 discusses the simulation script and section 7.1.2 gives information on the particulars of the simulation experiments that were conducted.

7.1.1 The Simulation Script

The network representation contained within the simulation script is illustrated in figure 7.1 and is almost identical to the network setup presented in section 6.1 and is related to Scenario (B) in Schwefel *et al.* [74].

The topology consists of two main nodes which represent the edge routers I and E respectively. Nodes I and E are connected by two simplex links with capacities μ_r and $\mu_{r'}$ (bits/s) respectively with latencies τ_r and $\tau_{r'}$ (seconds) respectively and with buffer sizes K_r and $K_{r'}$ (packets) respectively. There is a separate pool of access nodes for each main node and a duplex link of capacity μ_{acc} (bits/s) and zero latency connects each access node to its corresponding main node.

[1] Available from `http://www.isi.edu/nsnam/ns/`



access nodes
TCP
1
2
X
μ_{acc} duplex
connection arrival
TCP
μ_r simplex, τ_r, K_r
$\mu_{r'}$ simplex, $\tau_{r'}$, $K_{r'}$
I
E
main node
main node
Sink
Sink
forward
backward

Figure 7.1: The simulation setup for TCP connections transmitting in the forward direction. The setup is the same for connections transmitting in the backward direction.

TCP connections arrive at rate ν_r (connections/s) to node I. Each connection originates at an access node and transmits a single file after which the connection terminates. Data packets are transmitted from the access node to node I and via the forward simplex link to the TCP sink at node E. The corresponding ACK packets travel from E to I over the backward simplex link and finally to the access node. The limit on the transmission speed of the access link is imposed at the TCP sender whereas in section 6.1 the limit is imposed at the TCP receiver. Each access node has sufficient buffer space so that packets are only lost at the main nodes. Only one TCP connection can be active at a time on an access node. A similar scenario holds for connections arriving to node E.

7.1.2 Information on the Simulation Experiments

Two sets (denoted by set 1 and set 2) of simulation experiments were conducted. They investigate respectively (1) the impact of the TCP connection request rates ν_r and $\nu_{r'}$ and (2) the simplex link buffer sizes K_r and $K_{r'}$ on the performance metrics of the network. Both sets of experiments were conducted for file sizes of average lengths 30, 60 and 120 packets, and a total of 600 simulations were conducted. Table 7.1 shows the values of the other network parameters.

The inter-arrival times between connections were exponentially distributed, and the file sizes were distributed according to the Pareto distribution of section 3.4. See [78] for results for exponentially distributed file sizes.

Each simulation experiment simulated $2x$ hours of network time. To ensure that transient effects do not affect the results, simulation data were recorded from the last x hours only and only from connections that started and terminated during the last x hours. Because we are using the Pareto

μ_{acc}	128 kbps
μ_r, $\mu_{r'}$	2 Mb/s
τ_r, $\tau_{r'}$	100 msec
p_{usr}	1540×8 bits
p_{ack}	40 bytes
w_{rx}	64 packets

Table 7.1: The values of the other network parameters.

distribution for the file lengths, it is possible that the longest connections started in the first x hours resulting in biased results. We acknowledge that this could be a problem. Table 7.2 shows the values chosen for x which depend on the average of the file sizes of the simulations.

file size (packets)	$2x$ (hours)
30	2
60	4
120	8

Table 7.2: The network times for simulations with different average file sizes.

A doubling of the average file size implies that the connection request rates must be halved to achieve approximately the same loads on the routers. The argument follows that the network time being simulated must be doubled in order to record roughly the same number of connections for equal loads.

A sufficient number of simulations were run to insure a 95% confidence interval on the average number of connections active at the main nodes. (See appendix C.1 for information on acquiring confidence intervals.) Each data point in the plots that follow represents the average of ten simulations.

The performance metrics investigated are summarized in table 7.3. Table 7.3 also shows the parameter against which the corresponding performance metric is plotted in the results for the two sets of simulation experiments. ρ_r and $\rho_{r'}$ are the loads on the forward and backward simplex links r and r' respectively. Each metric was calculated as the average of the corresponding metrics in the forward and backward directions respectively. For example T is the average of the throughput T_r and $T_{r'}$ per TCP connection of connections transmitting over link r and r' respectively. In order to calculate q and C, samples of the queue lengths and the number of active connections were recorded every 10 milliseconds. T was calculated as the total number of packets sent (not including retransmitted packets) divided by the total time that the connections were active. This is in accordance with definition 1 in chapter 4. The loss probability e was calculated as the total number of data packets lost divided by the total number of data packets sent. The average congestion window size w was calculated as the average RTT divided by the average number of

metric	set 1 vs	set 2 vs
main node loss probability e	ρ_r, $\rho_{r'}$	K_r, $K_{r'}$
main node average queue length q (not including the packet in service)	ρ_r, $\rho_{r'}$	K_r, $K_{r'}$
average number C of active connections	ρ_r, $\rho_{r'}$	K_r, $K_{r'}$
average TCP window size w	ρ_r, $\rho_{r'}$	K_r, $K_{r'}$
load ρ on the simplex links	ν_r, $\nu_{r'}$	K_r, $K_{r'}$
throughput T per TCP connection	ν_r, $\nu_{r'}$	K_r, $K_{r'}$

Table 7.3: The performance metrics investigated and the metrics against which they are plotted.

packets transmitted per second per connection where the RTT was approximated with equation (25). The loads ρ_r and $\rho_{r'}$ were approximated by equation (34).

7.2 Applicability and Usage of the TCP Models

The single source models of chapter 4 are only able to calculate the throughput T and require input parameters values that must be obtained from either TCP traces or, as in our case, from simulation experiments. Therefore they can only be used after simulation experiments have been conducted. They may, however, be useful for estimating a lower bound for T if upper bounds of, for example, the packet loss probability and the average RTT in the network are known. For small data transfers with little or no packet loss, we expect these models to over-estimate T. We thus place the limit T_{max} as redefined in equation (63) on their throughput predictions. As the average file size increases, we expect these models to give more accurate throughput results as most of them model bulk data transfers.

For the TCP-Engset model, the number N of sources was assigned as the average number C of connections obtained from the corresponding simulation. The average OFF time Z of a TCP source was set to

$$Z = \frac{C}{\nu} - \frac{d}{T} \tag{67}$$

where ν is the TCP connection request rate, T is the throughput per TCP connection obtained from the corresponding simulation and $d = \varphi/(p_{\text{usr}} - p_{\text{ack}})$ is the average number of packets per file. The first term on the right hand side of equation (67) represents the amount of time since the start of a transmission by one of the C sources and the start of the next transmission by the same source. The second term is the total time to transmit a file. The rest of the input parameters were the same as for the simulation experiments.

Also for the TCP-NBurst model, the number N of sources was assigned as the average number C of connections obtained from the corresponding simulation. The average OFF time Z was assigned as for the TCP-Engset model. Instead of using the modified N-Burst MMPP in the

TCP-NBurst model, we implemented it with the N-Exp MMPP. Although it is not difficult to implement TCP-NBurst with the modified N-Burst MMPP, the effort required to implement it efficiently (see Schwefel [72]) is beyond the scope of this thesis.

The TCP-AK models of chapter 6 are more versatile as they represent a more complex network setup than the models of chapter 4. Furthermore, the input parameters to the TCP-AK models are normally known to network operators, and are the same as the input parameters to our simulations which are based on the same network setup. It is also possible to integrate the single source models of chapter 4 into the TCP-AK1 and TCP-AK1ext models resulting in hybrid models that may yield better predictions of the performance metrics of the network. For the TCP-AK1 and TCP-AK1ext models the attraction factor α was set to one as this feature is not present in the simulations.

Although the TCP-AK1ext and TCP-AK2ext models were extended with the N-Burst/M/1/K and N-Exp/M/1/K queues, results for the models are shown for the N-Exp/M/1/K queue only as it produced better results.

All of the models presented in this thesis are computationally tractable. Execution times are typically less than 10 microseconds on an AMD Athlon XP1800+ processor. For the models that implement the N-Exp/M/1/K queue, execution times are low for small N, but increase linearly with increasing N. Execution times can be improved significantly by using the Math Kernel Library[2] developed by Intel for processors that implement the SSE and/or SSE2 instruction sets such as the Intel Pentium II and later processors and the AMD Duron and Athlon processors.

In summary, all the TCP models consist of two inter-related submodels, a submodel for TCP and a submodel for the network setup. For the models of chapter 4, the network submodel was omitted in the model specification and incorporated as part of the input parameters. These input parameters are not necessarily available to network operators. For the TCP-Engset and TCP-NBurst models of chapter 5 and the TCP-AK models of chapter 6, the network submodel is included in the model specification and the models take input parameters values that are likely to be available to network operators. Table 7.4 gives a summary of the important TCP features included in the TCP models.

7.3 Applicability of the M/M/1/K and SM/M/1/K Queues

Apart from evaluating the TCP models it is also necessary to evaluate the M/M/1/K , N-Burst/M/1/K and N-Exp/M/1/K queues by comparing their predictions of the loss probability e and the average queue length q against the simulation results. The queues predict e and q given the input parameters in table 7.5. Although the queues do not model TCP explicitly, they do model a packet arrival process to the main node. The M/M/1/K queue assumes a Poisson arrival process while the N-Burst/M/1/K and N-Exp/M/1/K queues use MMPPs in the form of N ON/OFF

[2]Available from http://www.intel.com/software/products/perflib/index.htm

	CE	SS	CA	TO	TD
TCP-FLOYD			√	√	
TCP-OTT			√	√	
TCP-UMass1			√	√	
TCP-UMass2			√	√	√
TCP-UWash	√	√	√	√	√
TCP-UMass3			√	single	√
TCP-Hungary			√		√
TCP-Uppsala		√		√	
TCP-Engset			√		√
TCP-NBurst			√		√
TCP-AK1			√		√
TCP-AK1ext	√	√			
TCP-AK2		√	√	√	√
TCP-AK2ext	√	√	√	√	√

Table 7.4: The TCP features modelled by the various TCP models. CE indicates connection establishment.

	M/M/1/K	N-Burst/M/1/K	N-Exp/M/1/K
the average packet arrival rate λ	√	√	√
the average packet service rate μ	√	√	√
the buffer size K	√	√	√
the number N of sources		$N = 3$	√
the OFF time Z of a source		√	√
the ON time U of a source		√	√

Table 7.5: The input parameters for the M/M/1/K , N-Burst/M/1/K and N-Exp/M/1/K queues.

sources. In order to investigate the accuracy of the queues in predicting e and q, we included their predictions in the results where their respective input parameters were calculated from the simulation results. (The execution times of the N-Burst/M/1/K queue for a larger number of sources and for a larger order TPT distributions were excessive.)

7.4 Summary of the Results

It is known (see Crovella and Bestavros [16], Leland *et al.* [39] and Paxson and Floyd [66]) that TCP traffic is bursty causing the loss probability e and average queue length q at a router to be higher than predicted by simple Poisson models such as, for example, the M/M/1/K queue. A surprising result from the simulation experiments is that e and q are lower than expected as these values are closer to the predictions of the M/M/1/K queue than the predictions of the more realistic N-Exp/M/1/K queue.

A possible reason for this is that the limit on the transmission speed of the access links causes the

TCP traffic to be better behaved than when no limit is imposed. Instead of a window of data being offered simultaneously to a main node, the access link with limited capacity has a smoothing effect on the arrival of the packets to the main node. To test this hypothesis we ran simulations with an average file length of 30 packets for increasing connection request rates for the somewhat extreme case where the access links and the simplex links have the same capacities of 2 Mbits/s. In figure 7.2 we see that the M/M/1/K queue is inadequate for predicting e and q. For this situation the N-Exp/M/1/K queue gives better results and the TCP-AK2ext model gives better results than the TCP-AK2 model as shown in figure 7.3.

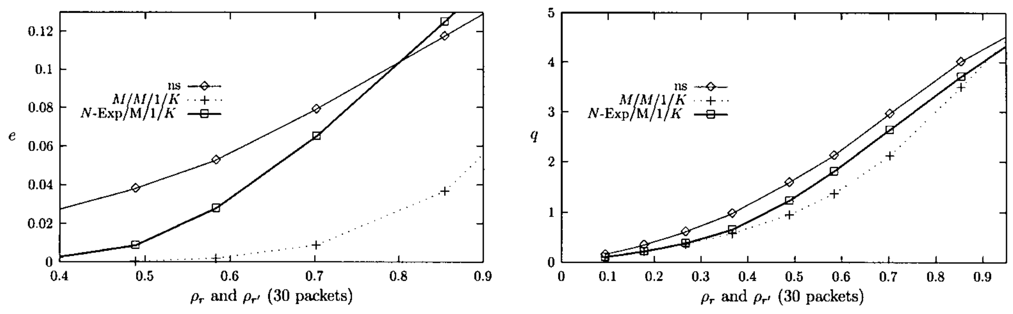


Figure 7.2: The loss probability e and average queue length q at the main nodes *vs* the loads ρ_r and $\rho_{r'}$ for an average file size of 30 packets: access link capacity 2 Mbits/s.

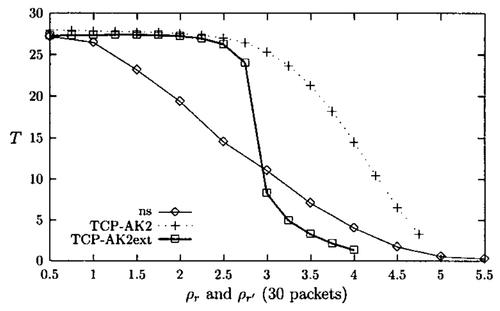


Figure 7.3: Throughput T (packets/s) per TCP connection *vs* the connection request rates ν_r and $\nu_{r'}$ for an average file size of 30 packets: access link capacity 2 Mbits/s.

For scenarios where the M/M/1/K and N-Exp/M/1/K queues are inadequate to predict e and q one can use the SM/M/1/K queue with a custom MMPP fitted to TCP traces as explained in appendix A.1.

We note that the results do not vary significantly for different file sizes. Throughput predictions tend to improve with larger file sizes. It may be more useful in future work to investigate the impact of different packet sizes and user access links with different capacities on the performance metrics of the network.

For the parameter values investigated table 7.6 shows the models which gave the best results compared to the simulation results. The TCP-AK1(Uppsala) and TCP-AK2 models gave the best overall results where the TCP-AK1(Uppsala) model is the TCP-AK1 model that uses the throughput formula of the TCP-Uppsala model.

	set 1				set 2			
	T	ρ	C	w	T	ρ	C	w
TCP-Uppsala	√				√			
TCP-Engset	√				√			
TCP-AK1		√				√		
TCP-AK1ext		√	√			√	√	
TCP-AK1(Uppsala)	√	√	√		√	√	√	
TCP-AK2	√	√	√	√	√	√		√
TCP-AK2ext		√		√		√		√

Table 7.6: The models that give the best predictions for the various performance metrics.

7.5 Results: The Impact of the Connection Request Rates

In this set of experiments the connection request rates ν_r and $\nu_{r'}$ were increased until the throughput per TCP connection saturated when the loads on the main nodes exceeded 1.0. The simplex link buffer sizes K_r and $K_{r'}$ were kept constant at 10 (packets), and the rest of the parameters are given in table 7.1. The results are presented in figures 7.4 to 7.13.

In figure 7.4 we see that the loss probabilities produced by the M/M/1/K queue are sufficiently accurate. This is a surprising result as we expect the simulation loss rates to be higher as a result of bursty TCP traffic.

A similar observation can be made in figure 7.5 where the average queue lengths q are close to the M/M/1/K average queue lengths. We expected q to be higher than the queue lengths predicted by the M/M/1/K queue, also as a result of the burstiness of the TCP traffic.

The throughput predictions of the single source models of chapter 4 are presented in figures 7.6 to 7.8. The plots for the TCP-UMass2 and TCP-UWash models nearly coincide. The TCP-Uppsala model gives the most accurate results. In figure 7.9 we see that of the two models of chapter 5, the TCP-Engset model gives the best throughput results.

Given that the TCP-Uppsala model gives the best throughput results of the single source models, we used its throughput formula and equation (61) to extend the TCP-AK1 model. We denote this new hybrid model by TCP-AK1(Uppsala). The throughput predictions of the TCP-AK models are presented in figure 7.10. The TCP-AK1ext model gives good results and the TCP-AK1(Uppsala) and TCP-AK2 models give the best results.

Figure 7.11 shows that all the TCP-AK models give good predictions for the load ρ. The TCP-AK1(Uppsala) model gives the best predictions for the average number C of connections as presented in figure 7.12. In figure 7.13 we see that the TCP-AK2 and TCP-AK2ext models give the best predictions for the average congestion window size w.

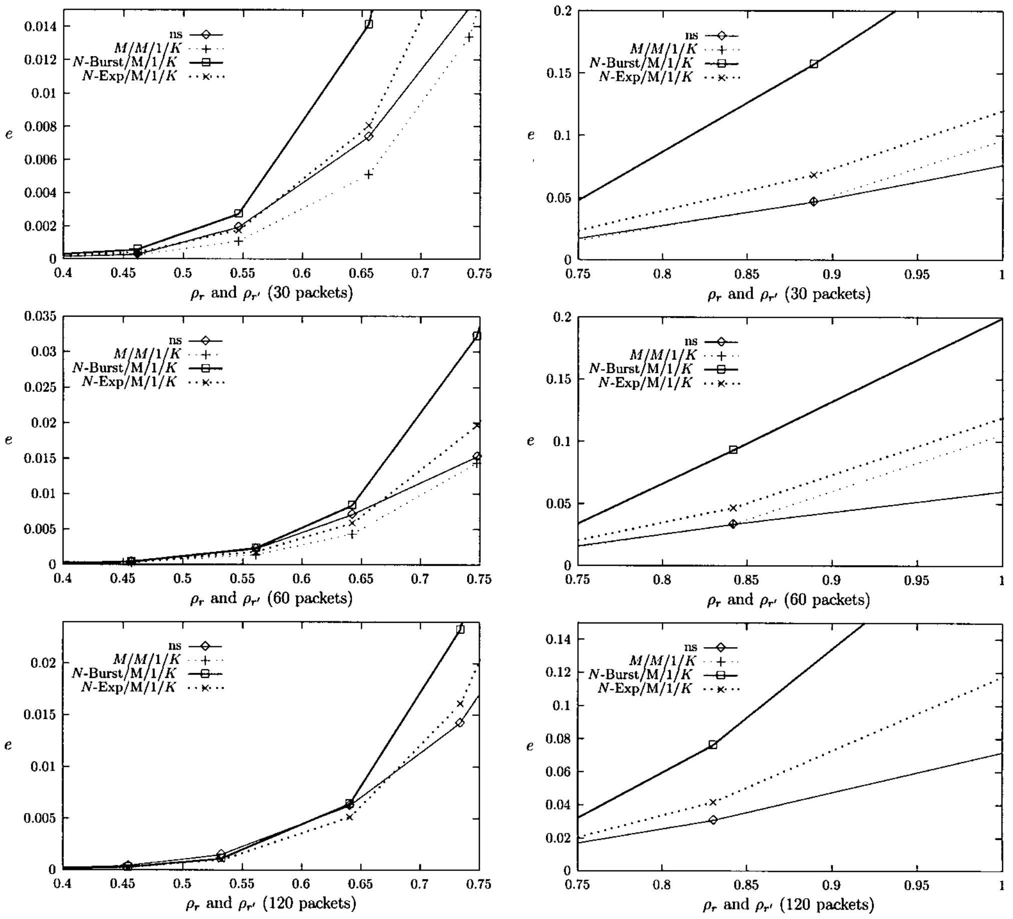


Figure 7.4: The loss probability e at the simplex links *vs* the loads ρ_r and $\rho_{r'}$ for average file sizes of 30, 60 and 120 packets respectively.

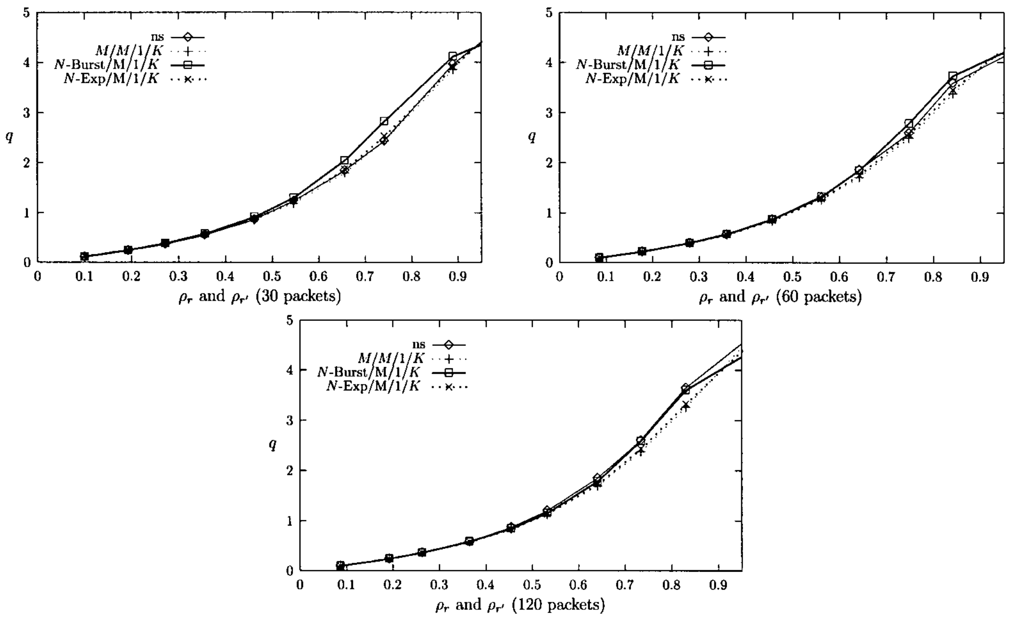


Figure 7.5: The average queue length q at the simplex links *vs* the loads ρ_r and $\rho_{r'}$ for average file sizes of 30, 60 and 120 packets respectively.

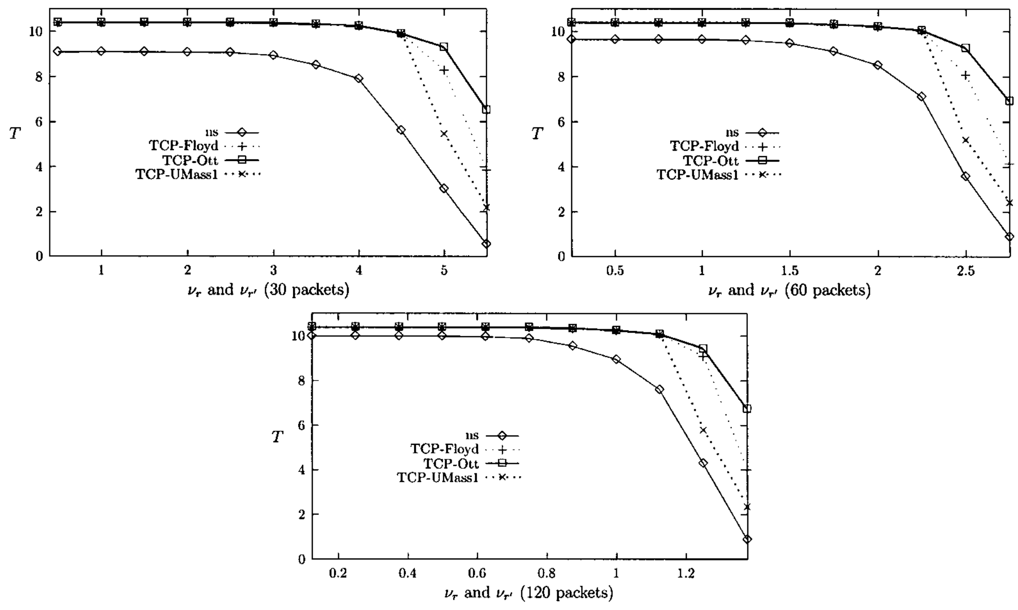


Figure 7.6: Throughput T (packets/s) per TCP connection *vs* the connection request rates ν_r and $\nu_{r'}$ for average file sizes of 30, 60 and 120 packets respectively.

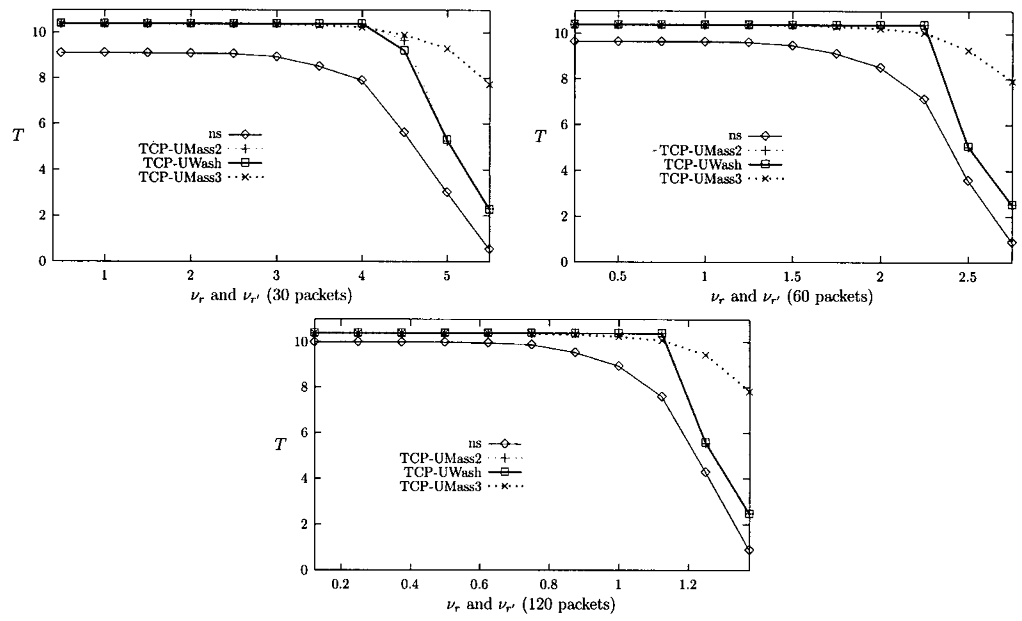


Figure 7.7: Throughput T (packets/s) per TCP connection *vs* the connection request rates ν_r and $\nu_{r'}$ for average file sizes of 30, 60 and 120 packets respectively. The TCP-UMass2 and TCP-UWash models nearly coincide.

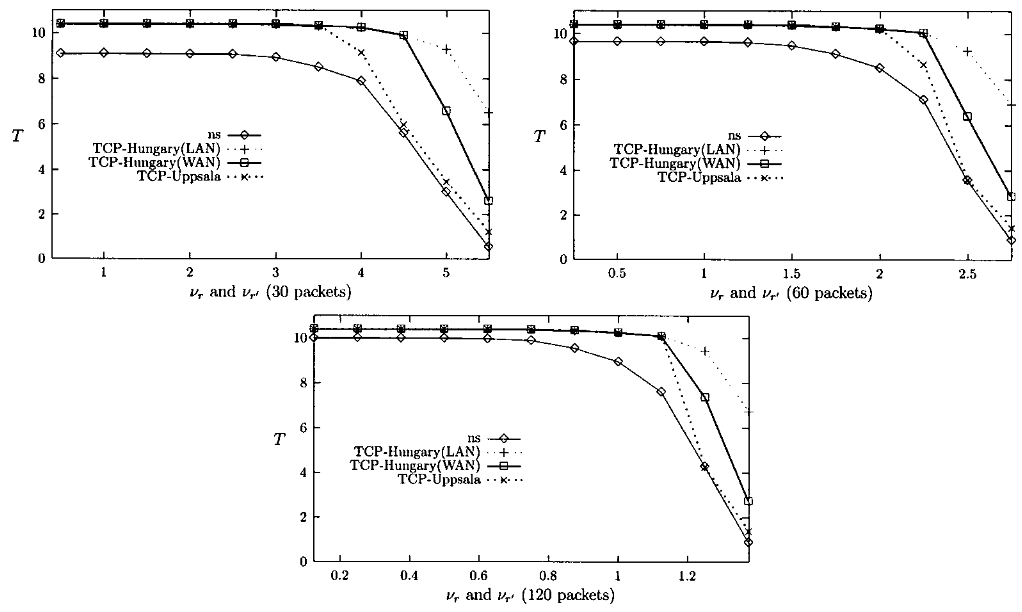


Figure 7.8: Throughput T (packets/s) per TCP connection *vs* the connection request rates ν_r and $\nu_{r'}$ for average file sizes of 30, 60 and 120 packets respectively.

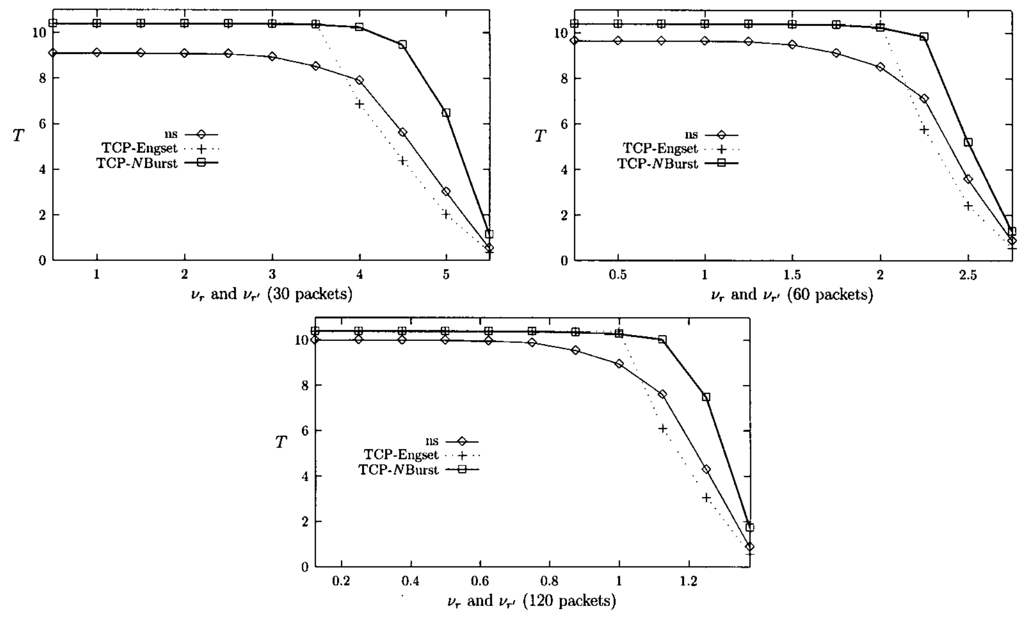


Figure 7.9: Throughput T (packets/s) per TCP connection *vs* the connection request rates ν_r and $\nu_{r'}$ for average file sizes of 30, 60 and 120 packets respectively.

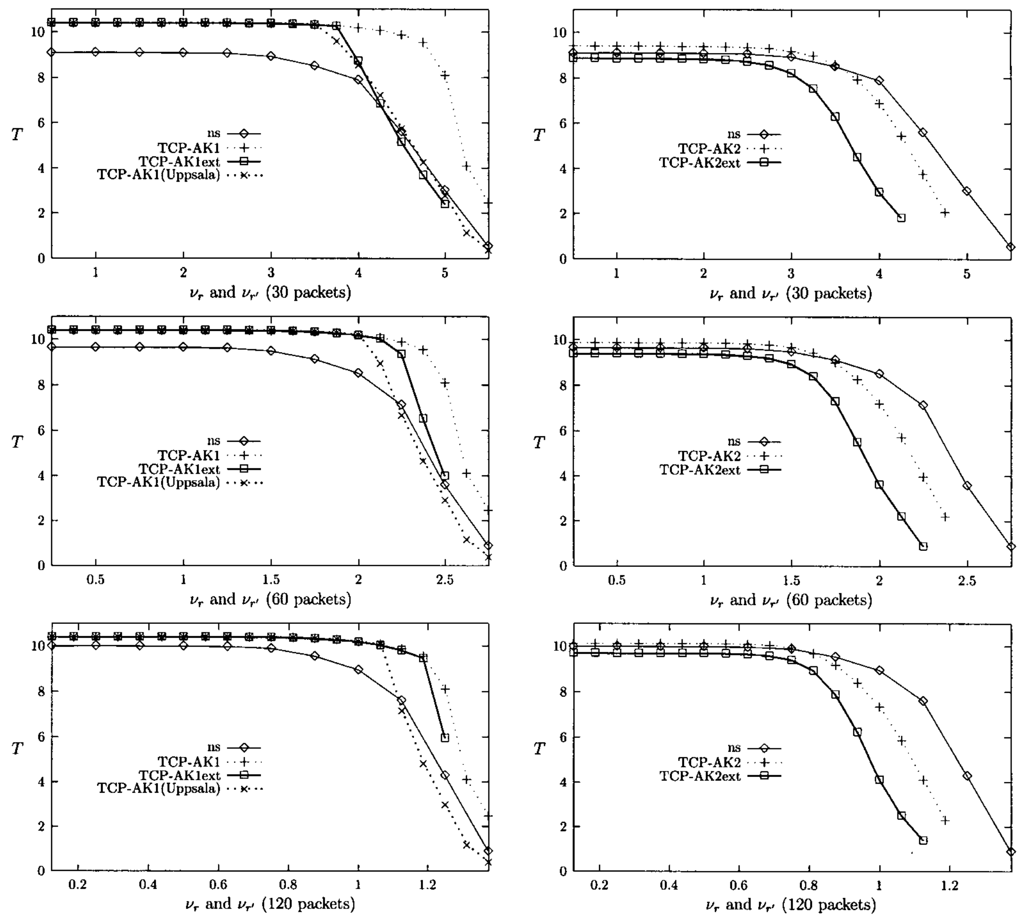


Figure 7.10: Throughput T (packets/s) per TCP connection *vs* the connection request rates ν_r and $\nu_{r'}$ for average file sizes of 30, 60 and 120 packets respectively.

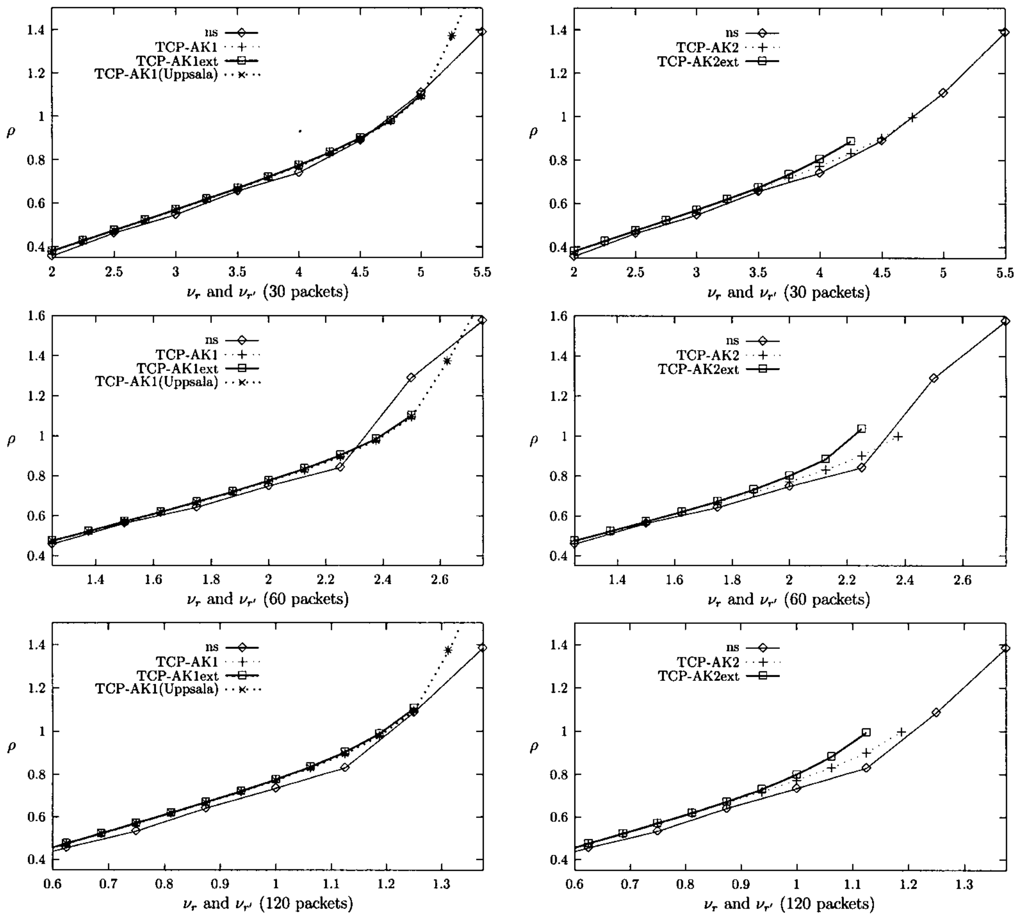


Figure 7.11: The load ρ on the simplex links *vs* the connection request rates ν_r and $\nu_{r'}$ for average file sizes of 30, 60 and 120 packets respectively.

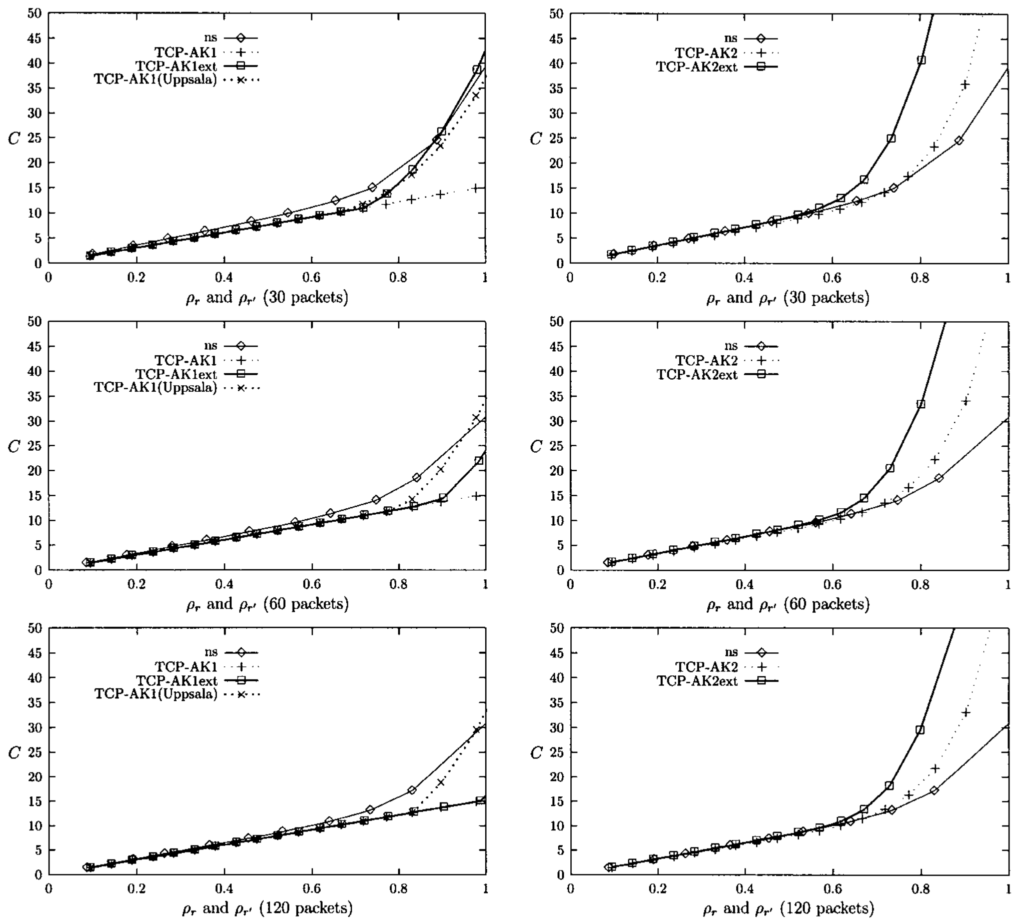


Figure 7.12: The average number C of active connections at the simplex links *vs* the loads ρ_r and $\rho_{r'}$ for average file sizes of 30, 60 and 120 packets respectively.

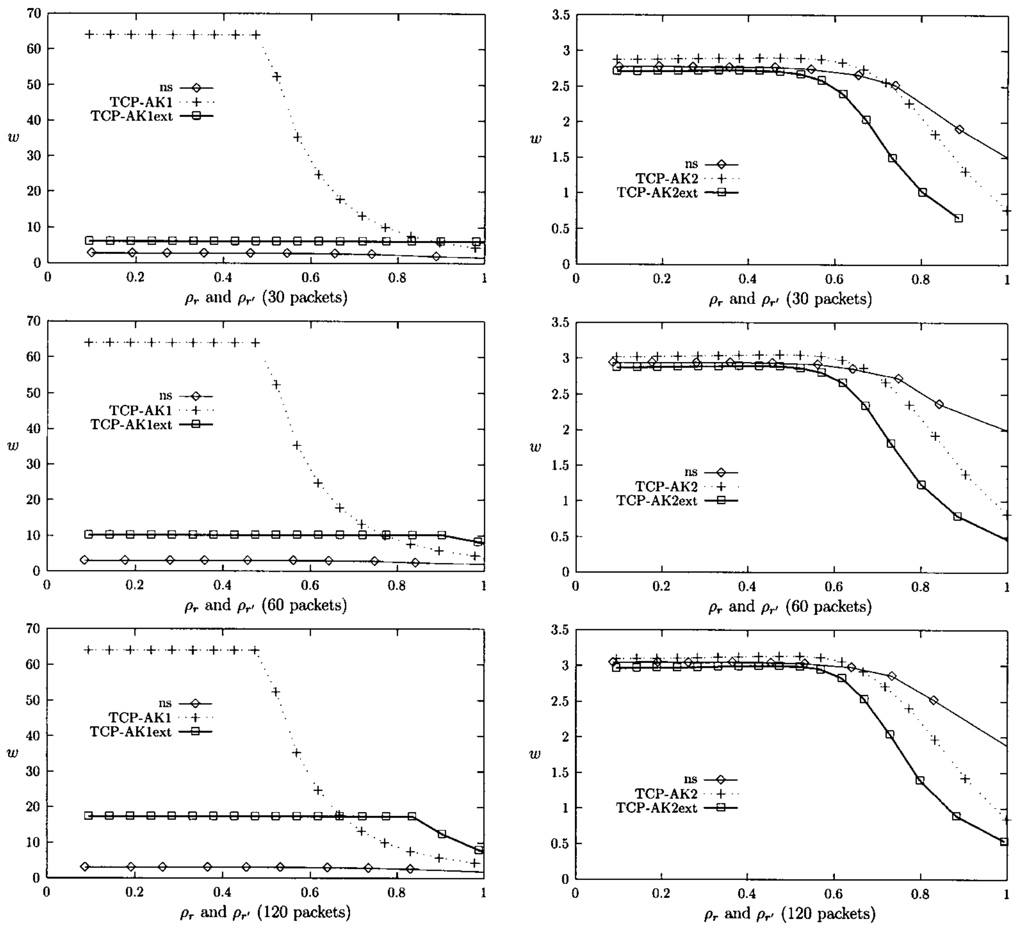


Figure 7.13: The average congestion window size w per connection *vs* the loads ρ_r and $\rho_{r'}$ for average file sizes of 30, 60 and 120 packets respectively.

7.6 Results: The Impact of the Simplex Link Buffer Sizes

In this set of experiments the simplex link buffer sizes K_r and $K_{r'}$ were increased from two to ten (packets). The TCP connection request rates ν_r and $\nu_{r'}$ were kept constant at 3, 1.5 and 0.75 (connections/s) for experiments with average file sizes of 30, 60 and 120 packets respectively. The TCP connection request rates were chosen from the results in section 7.5 as the rate values slightly smaller than the rate values corresponding to the start of the decline in the throughput per TCP connection. The rest of the parameters are given in table 7.1. Figures 7.14 to 7.23 show the results.

In figure 7.14 we see that all three the queues over-estimate the loss probability e. As with the set 1 experiments, we expected e to be higher due to the burstiness of TCP traffic. The M/M/1/K

queue gives the best results.

Figure 7.15 shows that all three queues give good predictions for the average queue length q. Once again we are surprised that the values of q are close to the M/M/1/K queue average queue lengths. The N-Exp/M/1/K queue gives the best results but only slightly better than the results of the M/M/1/K queue.

Figures 7.16 to 7.18 present the throughput predictions of the single source models of chapter 4. The plots for the TCP-UMass2 and TCP-UWash models nearly coincide. Once again it is the TCP-Uppsala model that gives the best throughput results. Of the two models in chapter 5 it is the TCP-Engset model that gives the best results as shown in figure 7.19.

The throughput predictions of the TCP-AK models are presented in figure 7.20. The TCP-AK1(Uppsala) and TCP-AK2 models give the best results.

In figure 7.21 we see that all of the TCP-AK models give good predictions for the load ρ for higher buffer sizes. Figure 7.22 shows that the TCP-AK1(Uppsala) model gives the best predictions for the average number C of connections and figure 7.23 shows that the TCP-AK2 and TCP-AK2ext models give the best predictions of the average congestion window size w.

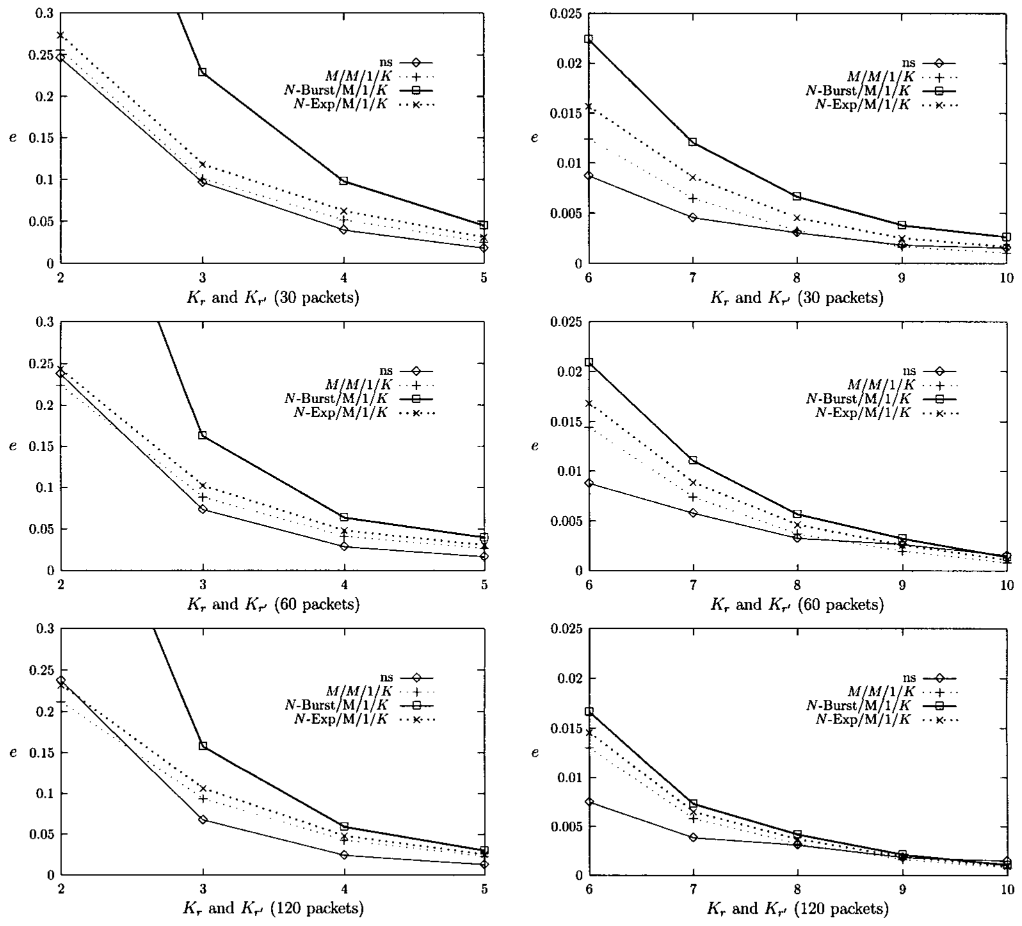


Figure 7.14: The loss probability e at the simplex links *vs* the buffer sizes K_r and $K_{r'}$ for average file sizes of 30, 60 and 120 packets respectively.

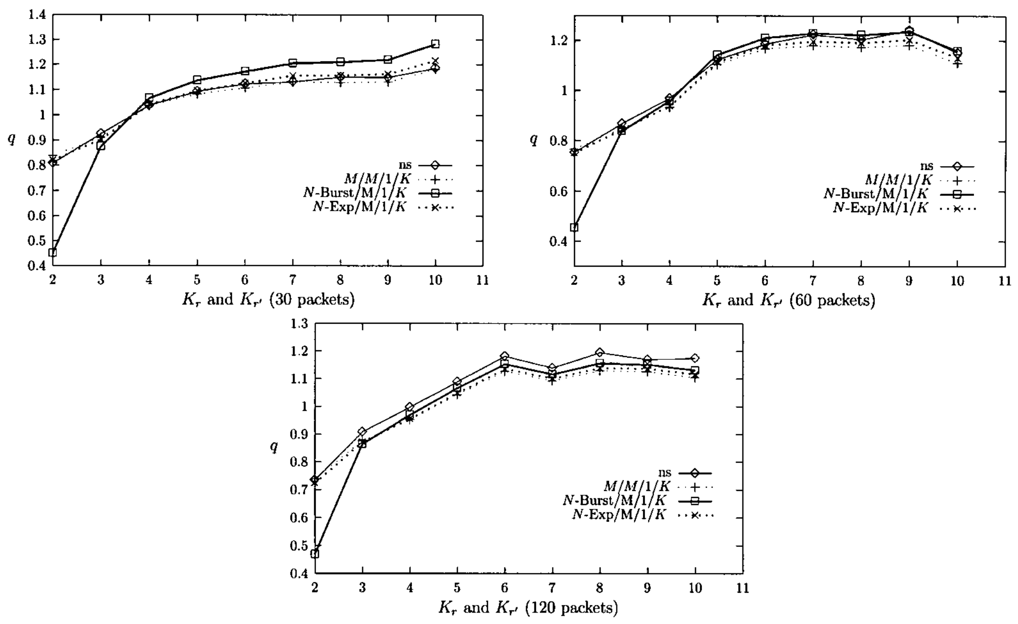


Figure 7.15: The average queue length q at the simplex links *vs* the buffer sizes K_r and $K_{r'}$ for average file sizes of 30, 60 and 120 packets respectively.

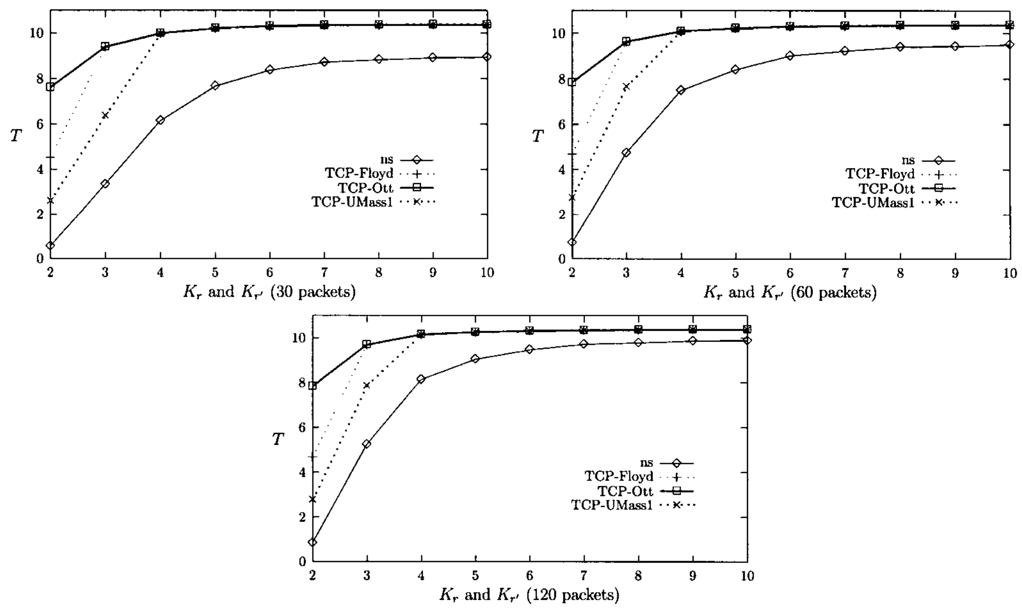


Figure 7.16: Throughput T (packets/s) per TCP connection *vs* the buffer sizes K_r and $K_{r'}$ for average file sizes of 30, 60 and 120 packets respectively.

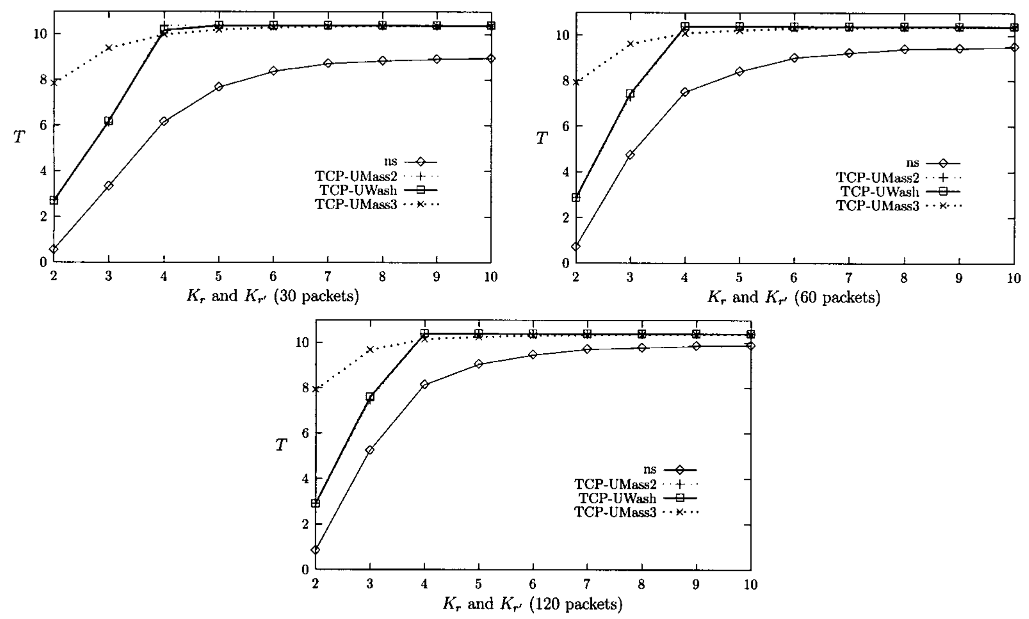


Figure 7.17: Throughput T (packets/s) per TCP connection *vs* the buffer sizes K_r and $K_{r'}$ for average file sizes of 30, 60 and 120 packets respectively. The TCP-UMass2 and TCP-UWash models nearly coincide.

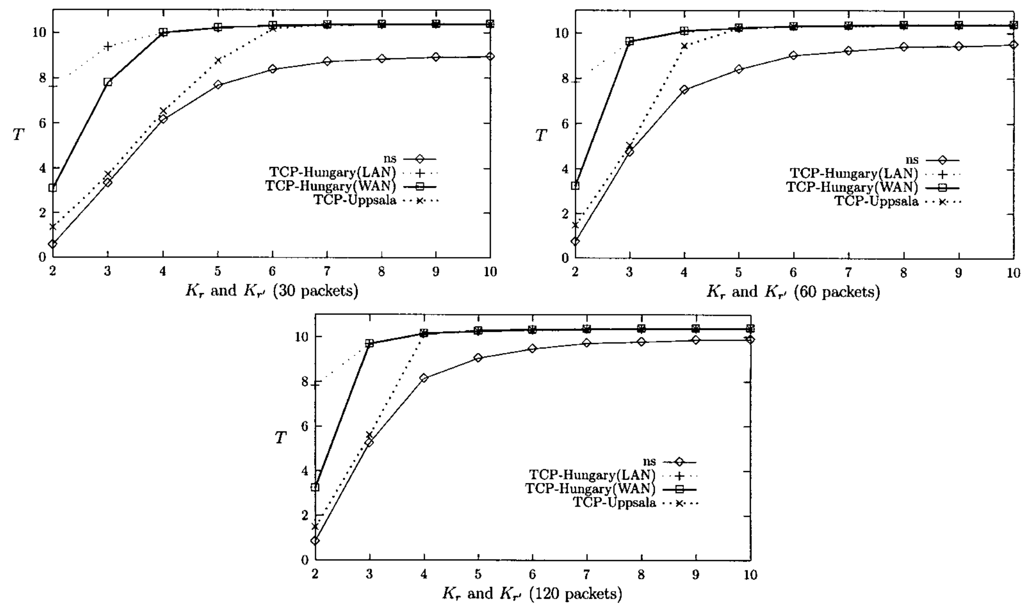


Figure 7.18: Throughput T (packets/s) per TCP connection *vs* the buffer sizes K_r and $K_{r'}$ for average file sizes of 30, 60 and 120 packets respectively.

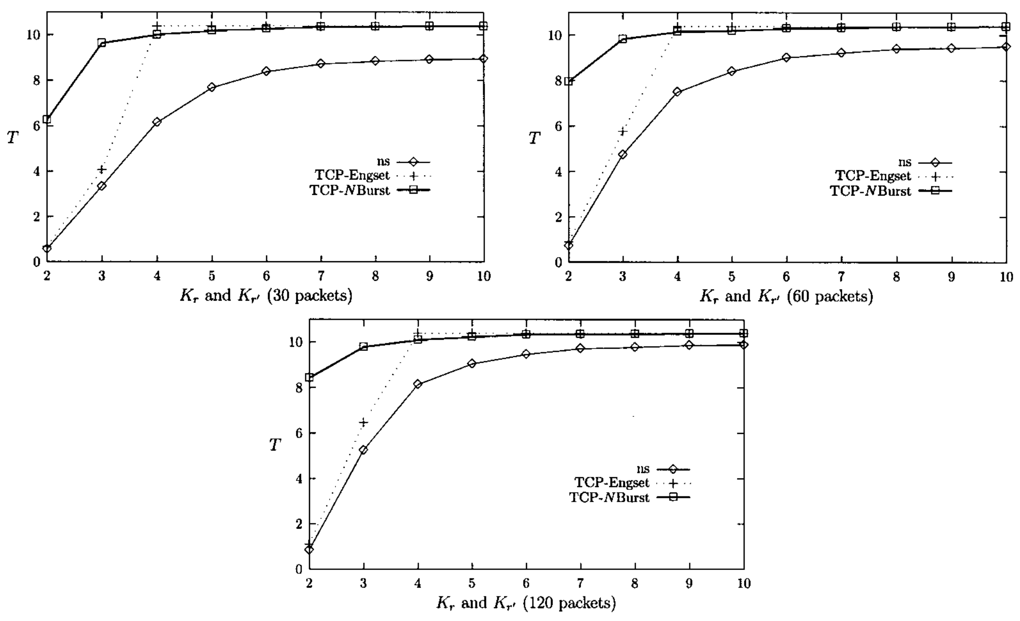


Figure 7.19: Throughput T (packets/s) per TCP connection *vs* the buffer sizes K_r and $K_{r'}$ for average file sizes of 30, 60 and 120 packets respectively.

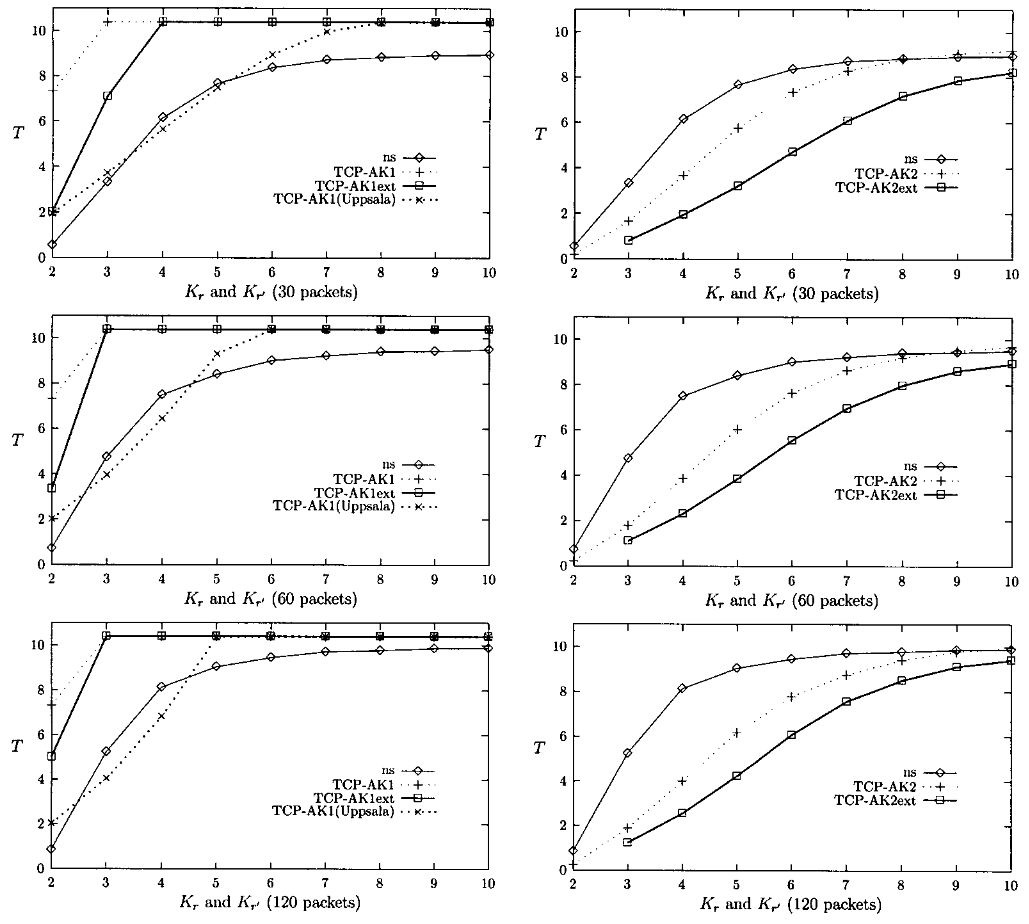


Figure 7.20: Throughput T (packets/s) per TCP connection *vs* the buffer sizes K_r and $K_{r'}$ for average file sizes of 30, 60 and 120 packets respectively.

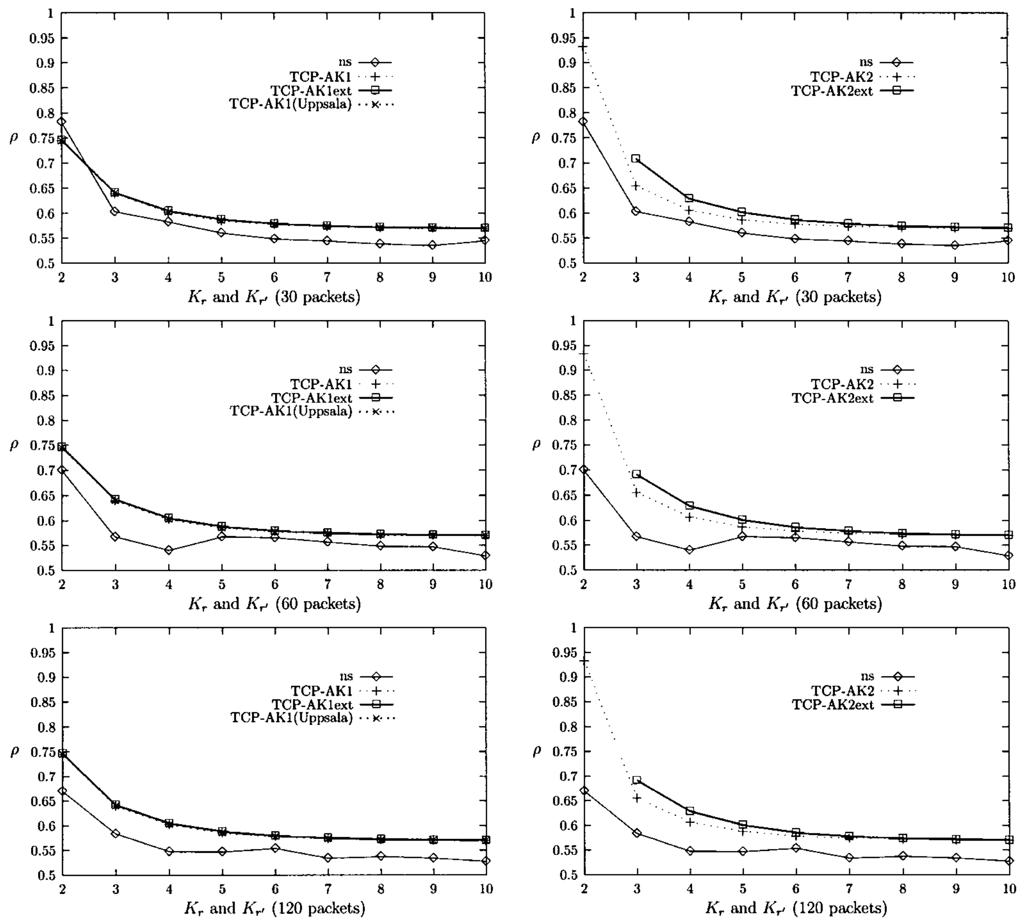


Figure 7.21: The load ρ on the simplex links *vs* the buffer sizes K_r and $K_{r'}$ for average file sizes of 30, 60 and 120 packets respectively.

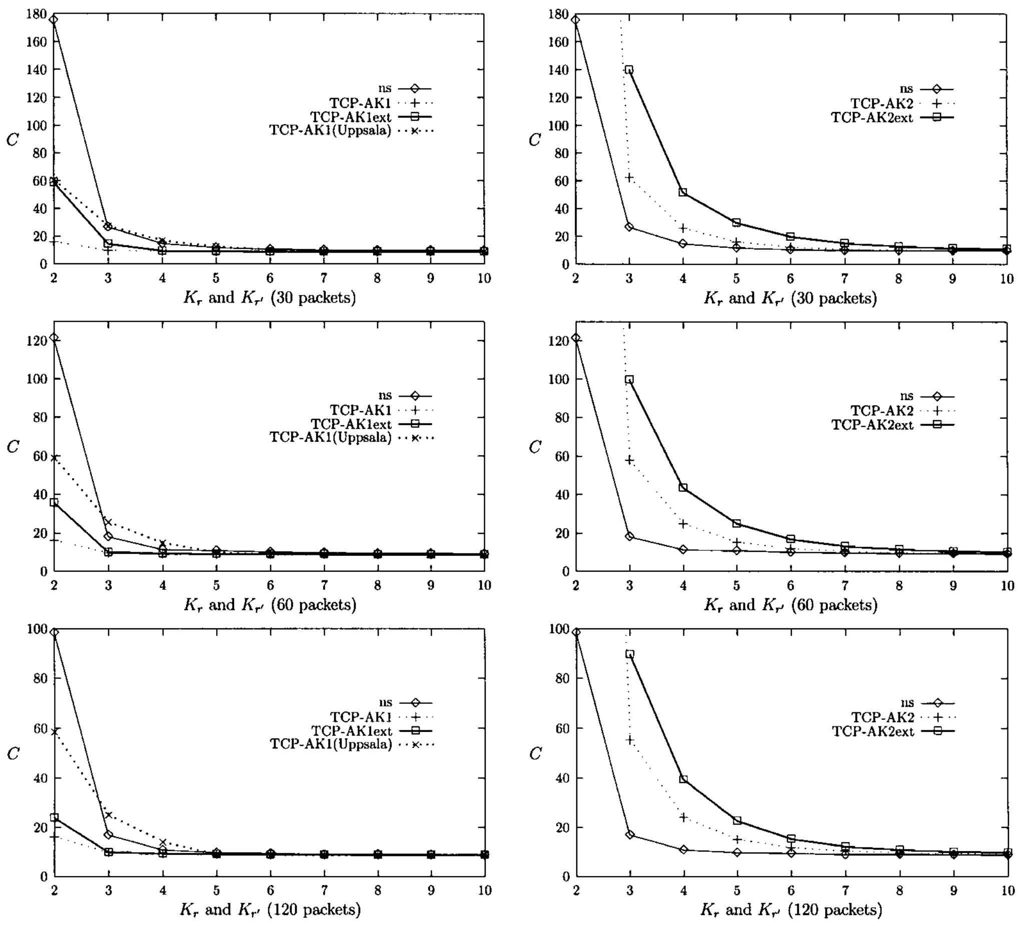


Figure 7.22: The average number C of active connections at the simplex links *vs* the buffer sizes K_r and $K_{r'}$ for average file sizes of 30, 60 and 120 packets respectively.

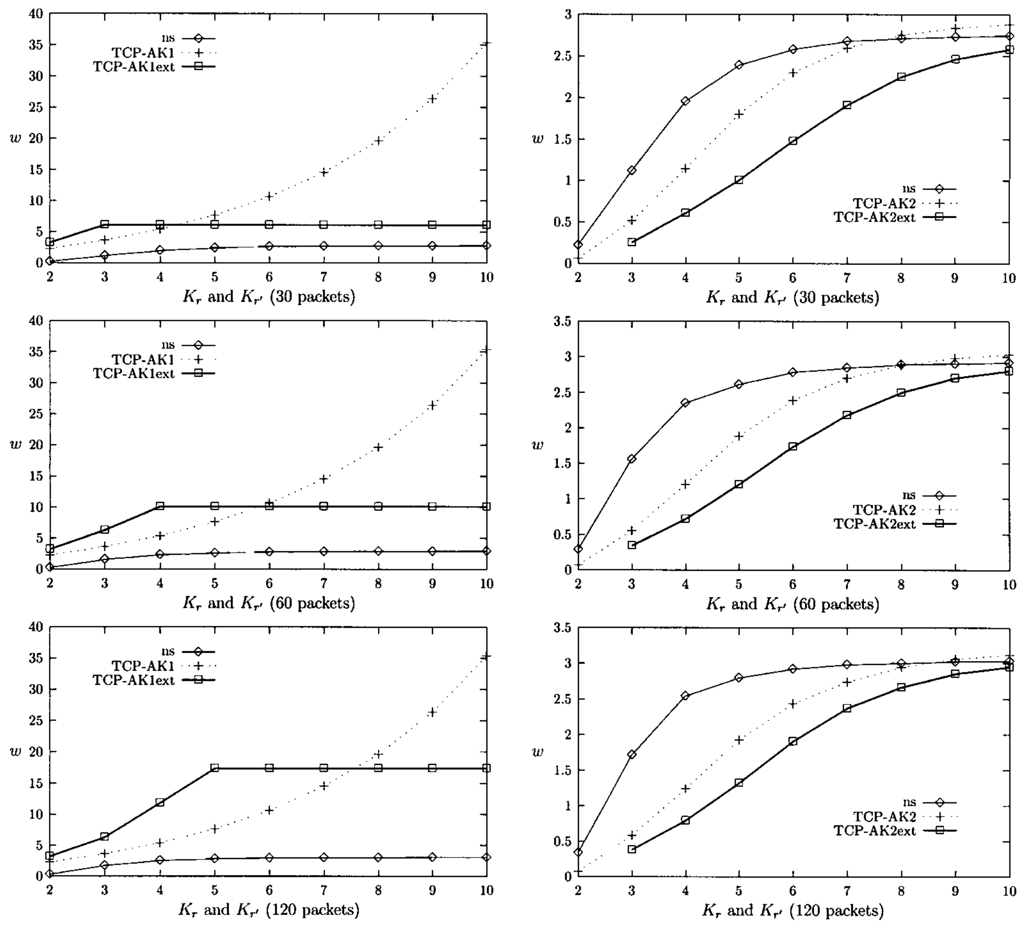


Figure 7.23: The average congestion window size w per connection *vs* the buffer sizes K_r and $K_{r'}$ for average file sizes of 30, 60 and 120 packets respectively.

7.7 Concluding Remarks

Owing to the complexity of the simulation script, the execution times of simulations with large traffic loads and long average file sizes are in excess of 8 hours per simulation on an AMD Athlon XP1800+ processor. Therefore, execution times can increase by a large factor when extending the simulation to represent larger networks. A method is proposed in appendix C.2 for extending the simulation script in such a way as to keep its increase in complexity and execution time to a minimum.

Chapter 8

Conclusion

This thesis investigated several tractable TCP models to find the ones most suitable to represent TCP traffic in MPLS networks. Two of these models were extended by implementing a more complex queue, the SM/M/1/K queue, which better describes the burstiness of TCP traffic than the M/M/1/K queue. The relevant TCP features included by these models were investigated (see table 7.4). The models were evaluated by comparing their throughput predictions versus results obtained from simulation experiments. Some models were also evaluated according to their predictions of performance metrics such as the average window size per TCP connection. It was found that some models show significantly better results than others in predicting different performance metrics (see table 7.6).

The simulation experiments revealed that for the network setup that was simulated the M/M/1/K queue is sufficient for modelling the traffic that arrives to an ELR. An example was given of a network setup where the M/M/1/K queue is not suitable to model the traffic arriving to the ELRs and where the more complex SM/M/1/K queue gives better results.

Based on our results of the network setup that was simulated, we conclude that some of the TCP models described in this thesis are sufficiently accurate to represent TCP traffic in MPLS networks.

We also presented a simple derivation of the $1/\sqrt{e}$ law for the TCP congestion window size where e is the packet loss probability.

8.1 Criticism and Future Work

The first point of criticism is that we did not examine measurements from real networks. We assumed that the *ns* simulations were sufficiently accurate to represent the network setup that was simulated. Future work might show whether this assumption was valid.

Another point of criticism is the simulation itself *i.e.* it is too complex in the sense that it takes too long and that, by extending the simulation script, its execution time will increase by a large factor. For future work, it could be insightful to simulate LSPs which consist of more than one link with additional traffics. To do this, one would have to extend the simulation in such a way that the increase in execution time is minimal. A way of doing this was described.

This thesis investigated the impact of different average file sizes on the performance metrics of the network. Future work may also investigate the impact of different packet sizes and user access links with different capacity sizes on the performance metrics of the network.

Appendix A

Additional Mathematical Background

This appendix gives additional mathematical background. A MMPP for which the corresponding matrices are determined by an empirical method is discussed, as well as the original Engset model.

A.1 MMPP LucHey: A Fitted Arrival Process

Instead of using a MMPP based on an analytic model such as the 1-Burst process, it is possible to construct a MMPP based on TCP traces. Assume we have data which capture the packet arrival rate to a router for each fixed interval t over a period of time. We can construct the LucHey MMPP (see Heyman and Lucantoni [29]) as follows.

Recall from section 3.5 the matrices $\mathbf{Q}$ and $\mathbf{L}$, and let μ denote the peak arrival rate of data packets. We choose the Poisson arrival rates $\{\lambda_i\}$ for the rate matrix $\mathbf{L}$ according to the following algorithm.

```
1. $N \leftarrow 1$
2. $\lambda_1 \leftarrow (\sqrt{1+\mu} - 1)^2$
3. $r \leftarrow \lambda_1 - 2\sqrt{\lambda_1}$
4. **if** $(r \leq 0)$ **stop**
5. $j \leftarrow N + 1$
6. $\lambda_j \leftarrow (\sqrt{1+r} - 1)^2$
7. $r \leftarrow \lambda_j - 2\sqrt{\lambda_j}$
```

```
8. $N \leftarrow j$
9. **if** $(r \leq 0)$ **stop**
   **else go to step** 5
```

The rate matrix **L** of the MMPP is thus N dimensional with rates $\{\lambda_i\}$ on the main diagonal. Let $\{x_i, \quad i = 1, 2, \ldots, T\}$ denote the data observation points. We associate x_i with phase j if

$$\lambda_j - 2\sqrt{\lambda_j} < x_i \leq \lambda_j + 2\sqrt{\lambda_j}.$$

Define

$$p_{ij} = \frac{\text{number of transitions from phases } i \text{ to } j}{\text{number of transitions out of phase } i}.$$

The infinitesimal generator matrix **Q** of the MMPP is constructed as follows

$$\begin{aligned} Q_{ij} &= tp_{ij} \quad i \neq j, \\ Q_{ii} &= t(p_{ii} - 1). \end{aligned}$$

When using the N-Burst MMPP of section 3.7 the state space can become very large as the number N of ON/OFF sources increases. Instead of using the N-Burst MMPP, one can simulate N ON/OFF sources and then, by using the procedure discussed in this section, represent the N-Burst MMPP by the LucHey MMPP.

A.2 The Engset Model

The Engset model (see Cohen [14], Cooper [15] and Heyman [31]) models a queuing system where N ON/OFF sources use a link which can handle at most s sources ($N > s$). The length of an ON period has distribution function $F(\cdot)$ with mean $1/f$ and the length of an OFF period has distribution function $G(\cdot)$ with mean $1/g$.

In order to analyse this network scenario, it is temporarily assumed that the ON and OFF times have exponential distributions with parameters f and g respectively, so that the number of active sources evolve according to a birth-and-death process with birth rates $\{\lambda_i\}$ and death rates $\{\mu_i\}$

$$\lambda_i = \begin{cases} (N-i)g & i = 0, 1, \ldots, s-1 \\ 0 & i = s \end{cases},$$

$$\mu_i = \begin{cases} 0 & j = 0 \\ jf & j = 1, 2, \ldots, s \end{cases}.$$

It is then shown that the equilibrium distribution $\{\pi_i\}$ of the number of active sources of this birth-and-death process is insensitive to the distributions of the ON and OFF times, as long as they have means $1/f$ and $1/g$ respectively. Therefore $\{\pi_i\}$ is also the equilibrium distribution for the original process.

Appendix B

TCP Models Not Considered

Numerous models of TCP for various different network setups have been reported in the literature. The large body of knowledge that exists on TCP makes it impossible to consider all the relevant material in this thesis.

For this thesis we chose the models which, in our opinion, are the most applicable to the MPLS network setup as described in sections 1.1 and 6.1. The following list contains TCP models that were not considered because they were either too complex or because they were not entirely applicable.

1. In Brown [11] a model is developed for the scenario where a fixed number of TCP connections share a resource with connections measuring different round trip times.
2. The TCP models in Ott *et al.* [48, 50, 51] were derived from the TCP-Ott model [60] to model respectively single and multiple TCP flows traversing a router which implements algorithms such as RED (Random Early Detection) and ERD (Early Random Drop).
3. The models presented in Heidemann *et al.* [28] and Mahdavi [44] are special cases of the more complex TCP-UWash model of section 4.5.
4. The model presented in Partridge and Shepard [63] models TCP traffic over a satellite link.
5. The model of Lakshman and Madhow [38] deals with TCP traffic that competes with real time traffic in a busy network.
6. In Altman *et al.* [3] a stochastic model is developed for TCP/IP where losses occur according to a random process.
7. Malouch and Liu [45] developed a model for TCP in networks with differentiated services and which implements the RED algorithm.
8. The model by Kumar [37] deals with several TCP versions on lossy links.

9. In Baccelli and Hong [7] the evolution of the TCP window size is represented in terms of Max-Plus algebra for the network setup where TCP traffic has to traverse several deterministic or random routers.
10. The models of Meo *et al.* [2, 13, 26] model TCP with a closed queuing network model.
11. A methodology for network capacity planning is presented in Roughan *et al.* [70] and a basic example is given by using the $1/\sqrt{e}$ law (see section 4.1) for the average TCP congestion window size where e is the packet loss probability.
12. The work in Altman *et al.* [4], which is mathematically complex, models TCP for the case where the RTT is dependent on the size of the congestion window.

Appendix C

Simulation

This appendix describes the method for finding confidence intervals when doing simulations and proposes a method for extending the *ns* simulation script described in section 7.1.1 that was used for the simulation experiments.

C.1 Acquiring Confidence Intervals

Assume the simulation was executed M times with different random number seed values and let $\mathbf{X}^{(i)}$ denote the observations for run i. The mean $\overline{X}_i$ of the observations of run i is

$$\overline{X}_i = \frac{1}{|\mathbf{X}^{(i)}|} \sum_{n=1}^{|\mathbf{X}^{(i)}|} X_n^{(i)}$$

with the mean $\overline{X}$ of these means

$$\overline{X} = \frac{1}{M} \sum_{i=1}^{M} \overline{X}_i$$

and an estimation s^2 of the variance of these means

$$s^2 = \frac{1}{M-1} \sum_{i=1}^{M} (\overline{X} - \overline{X}_i)^2.$$

Let $t_{M-1}(x)$ denote the value of Student's t distribution with $r = M - 1$ degrees of freedom at the point x

$$t_{M-1}(x) = \frac{\left(\frac{r}{r+x^2}\right)^{(1+r)/2}}{\sqrt{r}B(\frac{1}{2}r, \frac{1}{2})}$$

with $B(a, b)$ the beta function. $B(a, b)$ is calculated with the `beta` routine in [68]. Furthermore, for $0 < \alpha < 100$ let $\beta = 100 - \alpha$. A confidence interval of at least $\beta\%$ will be obtained if

$$M \geq \left(t_{M-1}(\alpha/2)s/\delta\right)^2$$

where $2\delta = 2\alpha\overline{X}$ is the width of the confidence interval.

C.2 Extending the Simulation Script

One might wish to extend the *ns* simulation script of section 7.1.1 in order to represent a more complex network setup. If for example we are interested in the same performance metrics as in chapter 7 for the two LSPs represented in figure 7.1 but the LSPs traverse several links with these links carrying additional traffics, then the simulation script can be extended to represent the network setup illustrated in figure C.1.

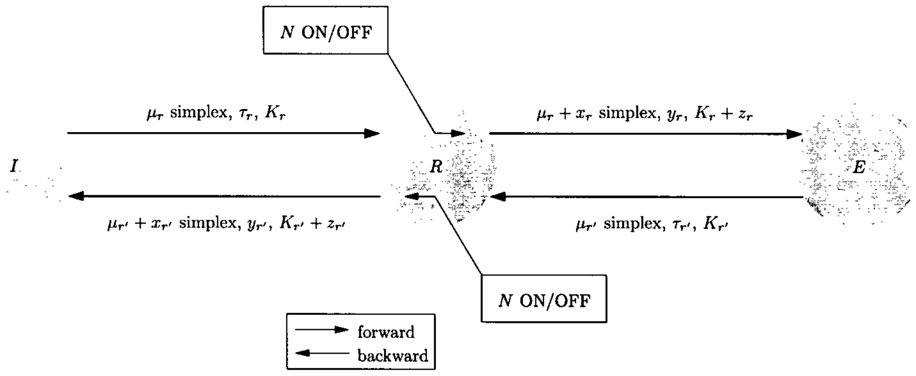

N ON/OFF
μ_r simplex, τ_r, K_r
$\mu_r + x_r$ simplex, y_r, $K_r + z_r$
I
R
E
$\mu_{r'} + x_{r'}$ simplex, $y_{r'}$, $K_{r'} + z_{r'}$
$\mu_{r'}$ simplex, $\tau_{r'}$, $K_{r'}$
N ON/OFF
forward
backward

Figure C.1: An extension of the network in figure 7.1.

In figure C.1 an extra node R was inserted between nodes I and E. The original forward link I-E now connects nodes I and R in the forward direction, and similarly the original backward link E-I. Nodes R and E are connected in the forward direction by a new link with capacity $\mu_r + x_r$ with latency y_r and with buffer size $K_r + z_r$. Nodes I and R are similarly connected. N ON/OFF sources at node R transmit in the forward direction to the nearest node E with an aggregate average bitrate that does not exceed x_r (bits/s). A similar description follows for the N ON/OFF sources at node R transmitting in the backward direction to the nearest node I.

A similar procedure can be followed to have more than one transit node between nodes I and E.

Another way of minimising execution times is to omit the TCP connections arriving to node E and compensate for the lack of data packet traffic at node E by decreasing its buffer size and link capacity. The user will then also have to compensate for the lack of ACK packet traffic at node I in a similar manner.

Appendix D

Delay in a Finite Single Server Queue

This appendix discusses properties of the waiting times in a single server queue which has K buffer spaces (including the server) and for which the processing times for packets have an exponential distribution with parameter μ. Let $\{a_k\}$ denote the distribution at packet arrival times of the number of packets in the queue.

D.1 The Waiting Time Probability Density Function

A packet that arrives to the queue with k packets in it will have a waiting time with probability density function $f_k(t)$ equal to the Erlang distribution with parameters $k+1$ and μ

$$f_k(t) = \frac{\mu(\mu t)^k \exp(-\mu t)}{k!}, \quad t > 0.$$

The waiting time for a packet arriving to the queue of arbitrary length has a probability density function $f(t)$ equal to

$$f(t) = \sum_{k=0}^{K-1} a_k f_k(t).$$

D.2 First and Second Moments and Variance of the Waiting Time

The first moment, or the average $\mathrm{E}(W)$ of the waiting time can be calculated from Little's Law $L = \lambda \mathrm{E}(W)$ where λ is the average arrival rate to the queue and L is the average queue length.

The i-th moment $\mathrm{E}(W^i)$ of the waiting time can be calculated with the Laplace Stieltjes transform $L(s)$ as explained in section 3.2

$$\begin{aligned} L(s) &= \int_0^\infty \exp(-st) f(t) dt \\ &= \int_0^\infty \exp(-st) \sum_{k=0}^{K-1} a_k \frac{\mu(\mu t)^k \exp(-\mu t)}{k!} dt \\ &= \sum_{k=0}^{K-1} a_k \frac{\mu^{k+1}}{k!} \underbrace{\int_0^\infty t^k \exp(-(s+\mu)t) dt}_{=k!/(s+\mu)^{k+1}} \\ &= \sum_{k=0}^{K-1} a_k \frac{\mu^{k+1}}{(s+\mu)^{k+1}}. \end{aligned}$$

The first and second moments are respectively

$$\begin{aligned} \mathrm{E}(W) &= -1 \left.\tfrac{dL(s)}{ds}\right|_{s=0} \\ &= \tfrac{1}{\mu} \textstyle\sum_{k=0}^{K-1} (k+1) a_k \end{aligned} \quad \text{and} \quad \begin{aligned} \mathrm{E}(W^2) &= \left.\tfrac{d^2 L(s)}{ds^2}\right|_{s=0} \\ &= \tfrac{1}{\mu^2} \textstyle\sum_{k=0}^{K-1} (k+1)(k+2) a_k. \end{aligned}$$

The variance $\mathrm{Var}(W)$ of the waiting time is

$$\mathrm{Var}(W) = \mathrm{E}(W^2) - \mathrm{E}(W)^2.$$

D.3 Example 1: The $M/M/1/K$ Queue

For the $M/M/1/K$ queue the distribution $\{a_k\}$ is equal to the stationary distribution $\{\pi_k\}$ of the queue length where

$$\pi_k = \begin{cases} \frac{1-\rho}{1-\rho^{K+1}} \rho^k & \rho \neq 1 \\ \frac{1}{K+1} & \rho = 1 \end{cases}.$$

Therefore, for $\rho = 1$

$$\begin{aligned} \mathrm{E}(W) &= \tfrac{1}{\mu} \textstyle\sum_{k=0}^{K-1} (k+1) \tfrac{1}{K+1} \\ &= \tfrac{1}{\mu} \tfrac{K}{2} \end{aligned} \quad \text{and} \quad \begin{aligned} \mathrm{E}(W^2) &= \tfrac{1}{\mu^2} \textstyle\sum_{k=0}^{K-1} (k+1)(k+2) \tfrac{1}{K+1} \\ &= \tfrac{K}{\mu^2} \tfrac{K+2}{3} \end{aligned}$$

and for $\rho \neq 1$

$$\begin{aligned} \mathrm{E}(W) &= \frac{1}{\mu} \sum_{k=0}^{K-1} (k+1) \frac{1-\rho}{1-\rho^{K+1}} \rho^k \\ &= \frac{1}{\mu} \frac{1}{1-\rho^{K+1}} \frac{1-(K+1)\rho^K + K\rho^{K+1}}{1-\rho} \end{aligned}$$

and

$$\begin{aligned}\mathrm{E}(W^2) &= \frac{1}{\mu^2}\sum_{k=0}^{K-1}(k+1)(k+2)\frac{1-\rho}{1-\rho^{K+1}}\rho^k \\ &= \frac{1}{\mu^2}\frac{1}{1-\rho^{K+1}}\frac{2-\rho^K[(K+1)(K+2)-2K(K+2)\rho+K(K+1)\rho^2]}{(1-\rho)^2}.\end{aligned}$$

D.4 Example 2: The $SM/M/1/K$ Queue

For the $SM/M/1/K$ queue, the distribution $\{a_k\}$ is given in equation (9). Therefore,

$$\begin{aligned}\mathrm{E}(W) &= \frac{1}{\mu}\sum_{k=0}^{K-1}(k+1)\frac{(\mathbf{a}\mathbf{R}^k+\mathbf{b}\mathbf{S}^{K-k})\mathbf{Le}}{\pi\mathbf{Le}} \\ &= \frac{1}{\mu}\left[\mathbf{a}\sum_{k=0}^{K-1}(k+1)\mathbf{R}^k+\mathbf{b}\sum_{k=0}^{K-1}(k+1)\mathbf{S}^{K-k}\right]\frac{\mathbf{Le}}{\pi\mathbf{Le}} \\ &= \frac{1}{\mu}\left[\mathbf{a}\mathbf{X}_1+\mathbf{b}\mathbf{X}_2\right]\frac{\mathbf{Le}}{\pi\mathbf{Le}}\end{aligned}$$

and

$$\begin{aligned}\mathrm{E}(W^2) &= \frac{1}{\mu^2}\sum_{k=0}^{K-1}(k+1)(k+2)\frac{(\mathbf{a}\mathbf{R}^k+\mathbf{b}\mathbf{S}^{K-k})\mathbf{Le}}{\pi\mathbf{Le}} \\ &= \frac{1}{\mu}\left[\mathbf{a}\sum_{k=0}^{K-1}(k+1)(k+2)\mathbf{R}^k+\mathbf{b}\sum_{k=0}^{K-1}2(k+1)\mathbf{S}^{K-k}+k\mathbf{S}^{K-k}+k^2\mathbf{S}^{K-k}\right]\frac{\mathbf{Le}}{\pi\mathbf{Le}} \\ &= \frac{1}{\mu}\left[\mathbf{a}\mathbf{X}_3+\mathbf{b}(2\mathbf{X}_2+\mathbf{X}_4+\mathbf{X}_5)\right]\frac{\mathbf{Le}}{\pi\mathbf{Le}}\end{aligned}$$

where

$$\begin{aligned}\mathbf{X}_1 &= \frac{\mathbf{I}-(K+1)\mathbf{R}^K+K\mathbf{R}^{K+1}}{(\mathbf{I}-\mathbf{R})^2} \\ \mathbf{X}_2 &= \frac{K\mathbf{S}-(K+1)\mathbf{S}^2+\mathbf{S}^{K+2}}{(\mathbf{I}-\mathbf{S})^2} \\ \mathbf{X}_3 &= \frac{2\mathbf{I}-\mathbf{R}^K[(K+1)(K+2)+2K(K+2)\mathbf{R}-K(K+1)\mathbf{R}^2]}{(1-\mathbf{R})^3} \\ \mathbf{X}_4 &= \frac{(K-1)\mathbf{S}^2-K\mathbf{S}^3+\mathbf{S}^{K+2}}{(\mathbf{I}-\mathbf{S})^2\mathbf{S}}\quad\left(=\frac{\mathbf{X}_2-K\mathbf{S}}{\mathbf{S}}\right) \\ \mathbf{X}_5 &= \frac{(K-1)^2\mathbf{S}-(2K^2-6K+5)\mathbf{S}^2+(K^2-2K+3)\mathbf{S}^3-(\mathbf{I}+\mathbf{S})\mathbf{S}^{K+1}}{(\mathbf{I}-\mathbf{S})^3}.\end{aligned}$$

The derivations of $\mathbf{X}_1$ and $\mathbf{X}_3$ are respectively similar to the derivations of $\mathrm{E}(W)$ and $\mathrm{E}(W^2)$ for the $M/M/1/K$ queue in the previous example.

Bibliography

[1] N. Akar and E. Arikan. A Numerically Efficient Method for the MAP/D/1/K Queue via Rational Approximations. *Queueing Systems: Theory and Applications*, 1996.

[2] E. Alessio, M. Garetto, R. L. Cigno, M. Meo, and M. A. Marsan. Analytical Estimation of Completion Times of Mixed NewReno and Tahoe TCP Connections over Single and Multiple Bottleneck Networks. In *IEEE Globecom 2001*, San Antonio, Texas, USA, November 2001.

[3] E. Altman, K. Avrachenkov, and C. Barakat. A Stochastic Model of TCP/IP with Stationary Random Losses. In *ACM SIGCOMM 2000*, pages 231–242, Stockholm, Sweden, August 2000.

[4] E. Altman, K. Avrachenkov, C. Barakat, and R. Núñez-Queija. TCP Modeling in the Presence of Nonlinear Window Growth. In *Seventeenth International Teletraffic Congress (ITC17)*, Salvador da Bahia, Brazil, December 2001.

[5] Å. Arvidsson and A.E. Krzesinski. A Model of a TCP Link. In *15th ITC Specialist Seminar on Internet Traffic Engineering and Traffic Management*, Würzburg, Germany, July 2002.

[6] Å. Arvidsson and A.E. Krzesinski. Design of Optimal Multi-Service MPLS Networks. In *Networks 2002*, pages 31–40, Munich, Germany, June 2002.

[7] F. Baccelli and D. Hong. TCP is Max-Plus Linear and what it Tells Us On its Throughput. In *ACM SIGCOMM 2000*, Stockholm, Sweden, August 2000.

[8] C. Blondia and F. Geerts. The Correlation Structure of the Output of an ATM Multiplexer. In *Modelling and Evaluation of ATM Networks*, pages 235–250, Ilkley, UK, July 1997.

[9] L.S. Brakmo, S.W. O'Malley, and L.L. Peterson. TCP Vegas: New Techniques for Congestion Detection and Avoidance. In *ACM SIGCOMM '94*, pages 24–35, London, UK, September 1994.

[10] L.S. Brakmo and L.L. Peterson. TCP Vegas: End to End Congestion Avoidance on a Global Internet. *IEEE Journal on Selected Areas in Communications*, 13(8):1465–1480, 1995.

[11] P. Brown. Resource Sharing of TCP Connections with Different Round Trip Times. In *IEEE INFOCOM 2000*, Tel-Aviv, Israel, March 2000.

[12] N. Cardwell, S. Savage, and T. Anderson. Modeling TCP Latency. In *IEEE INFOCOM (3)*, pages 1742–1751, Tel-Aviv, Israel, March 2000.

[13] C. Casetti and M. Meo. An Analytical Framework for the Performance Evaluation of TCP Reno Connections. In *Computer Networks*, volume 37, pages 669–682. Elsevier North-Holland Inc., New York, NY, USA, 2001.

[14] J.W. Cohen. The Generalized Engset Formula. *Phillips Telecommunications Review 18*, pages 158–170, 1957.

[15] R.B. Cooper. *Introduction to Queueing Theory.* North Holland, New York, second edition, 1981.

[16] M. Crovella and A. Bestavros. Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes. In *SIGMETRICS '96: The ACM International Conference on Measurement and Modeling of Computer Systems.*, Philadelphia, Pennsylvania, May 1996. Also in Performance Evaluation Review, May 1996, 24(1):160-169.

[17] B. Davie, P. Doolan, and Y. Rekhter. *Switching in IP Networks: IP Switching, Tag Switching, & Related Technologies.* Morgan Kaufman Publishers Inc., San Francisco, USA, 1998.

[18] A. Fekete, G. Vattay, and A. Veres. Improving the $1/\sqrt{p}$ Law for Single and Parallel TCP Flows. In *Seventeenth International Teletraffic Congress (ITC17)*, Salvador da Bahia, Brazil, December 2001.

[19] S. Floyd. Connections with Multiple Congested Gateways in Packet- Switched Networks Part 1: One-Way Traffic. *ACM Computer Communication Review*, 21(5):30–47, October 1991.

[20] S. Floyd. TCP and successive fast retransmits. Available from `ftp://ftp.ee.lbl.gov/papers`, February 1995.

[21] S. Floyd. The NewReno Modification to TCP's Fast Recovery Algorithm. *IETF RFC 2582*, April 1999.

[22] S. Floyd and K. Fall. Comparison of Tahoe, Reno and SACK TCP. Available from `http://www-nrg.ee.lbl.gov/nrg-papers.html`, 1995.

[23] S. Floyd and K. Fall. Promoting the Use of End-to-End Congestion Control in the Internet. *IEEE/ACM Transactions on Networking*, 7(4):458–472, August 1999.

[24] S. Floyd, M. Handley, and J. Padhye. A Comparison of Equation-Based and AIMD Congestion Control. Available from `http://www.aciri.org/tfrc/`, May 2000.

[25] G. Latouche and V. Ramaswami. *Introduction to Matrix Analytic Methods in Stochastic Models.* SIAM, 1999.

[26] M. Garetto, R.L. Cigno, M. Meo, and M. A. Marsan. A Detailed and Accurate Closed Queueing Network Model of Many Interacting TCP Flows. In *IEEE INFOCOM 2001*, pages 1706–1715, Anchorage, Alaska, April 2001.

[27] M. Greiner, M. Jobmann, and L. Lipsky. The Importance of Power-tail Distributions for Modeling Queueing Systems. *Operations Research*, 47(2), March 1999.

[28] J. Heidemann, K. Obraczka, and J. Touch. Modeling the Performance of HTTP Over Several Transport Protocols. *IEEE/ACM Transactions on Networking*, 5(5), October 1997.

[29] D. Heyman and D. Lucantoni. Modeling Multiple IP Traffic Streams With Rate Limits. In *Seventeenth International Teletraffic Congress (ITC17)*, Salvador da Bahia, Brazil, December 2001.

[30] D.P. Heyman, T.V. Lakshman, and Arnold L. Neidhardt. A New Method for Analysing Feedback-Based Protocols with Applications to Engineering Web Traffic over the Internet. In *ACM Sigmetrics*, pages 24–38, Seattle, Washington, USA, 1997.

[31] D.P. Heyman and M.J. Sobel. *Stochastic Models in Operations Research*, volume 1. McGraw-Hill, New York, 1982.

[32] V. Jacobson. Congestion Avoidance and Control. *Computer Communication Review*, 18(4):314–329, August 1988.

[33] V. Jacobson. Berkeley TCP Evolution from 4.3-Tahoe to 4.3-Reno. In *Eighteenth Internet Engineering Task Force*, University of British Colombia, Vancouver, BC, September 1990.

[34] V. Jacobson and R. Braden. TCP Extensions for Long-Delay Paths. *IETF RFC 1072*, October 1988.

[35] P. Karn and C. Partridge. Improving Round-Trip Time Estimates in Reliable Transport Protocols. *Computer Communications Review*, 17(5), August 1987.

[36] U.R. Krieger, V. Naoumov, and D. Wagner. Analysis of a Finite FIFO Buffer in an Advanced Packet-Switched Network. *IEICE Transactions on Communications*, E81-B(5), May 1998.

[37] A. Kumar. Comparative Performance Analysis of Versions of TCP in a Local Network with a Lossy Link. *IEEE/ACM Transactions on Networking*, 6(4):485–498, 1998.

[38] T. Lakshman and U. Madhow. The Performance of TCP/IP for Networks with High Bandwidth-Delay Products and Random Loss. *IEEE/ACM Transactions on Networking*, 5(3):336–350, July 1997.

[39] W. Leland, M. Taqqu, W. Willinger, and D.V. Wilson. On the Self-Similar Nature of Ethernet Traffic (Extended Version). *Proceedings IEEE/ACM Transactions on Networking*, 2:1–15, February 1994.

[40] L. Lipsky and P. Fiorini. Analytic Models Of Performance In Telecommunication Systems, Based On ON-OFF Traffic Sources With Self-Similar Behavior. In *7th International Conference on Telecommunications Systems*, March 1999.

[41] S.H. Low, L.L. Peterson, and L. Wang. Understanding TCP Vegas: a Duality Model. In *SIGMETRICS/Performance 2001*, pages 226–235, Cambridge, Massachusetts, USA, June 2001.

[42] D. M. Lucantoni. New Results on the Single Server Queue with a Batch Markovian Arrival Process. *Commun. Statist.-Stochastic Models*, 7(1):1–46, 1991.

[43] D. M. Lucantoni. The BMAP/G/1 Queue: A Tutorial. In *Models and Techniques for Performance Evaluation of Computer and Communications Systems, L. Donatiello and R. Nelson eds.*, pages 330–358. Springer Verlag, 1993.

[44] J. Mahdavi. TCP Performance Tuning. Available from `http://www.psc.edu/networking/tcptune/slides/`, April 1997.

[45] N.M. Malouch and Z. Liu. On Steady State Analysis of TCP in Networks with Differentiated Services. In *Seventeenth International Teletraffic Congress*, Brazil, September 2001.

[46] M. Mathis and J. Mahdavi. Forward Acknowledgement: Refining TCP Congestion Control. In *ACM SIGCOMM '96*, pages 281–291, Stanford University, California, USA, August 1996.

[47] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow. TCP Selective Acknowledgment Options. *IETF RFC 2018*, October 1996.

[48] M. Mathis, J. Semke, J. Mahdavi, and T. Ott. The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm. *Computer Communications Review*, 27(3), July 1997.

[49] S. MCanne and S. Floyd. ns-LBL Network Simulator, 1997. Available from `http://www.isi.edu/nsnam/ns/`.

[50] A. Misra and T. Ott. The Window Distribution of Idealized TCP Congestion Avoidance with Variable Packet Loss. In *IEEE Infocom '99*, New York, NY, USA, March 1999.

[51] A. Misra, T. Ott, and J. Baras. The Window Distribution of Multiple TCPs with Random Queues. In *IEEE GLOBECOM '99*, Rio de Janeiro, Brazil, December 1999.

[52] V. Misra, W.-B. Gong, and D. Towsley. Stochastic Differential Equation Modeling and Analysis of TCP-Windowsize Behavior. In *Performance '99*, Istanbul, Turkey, October 1999.

[53] J. Mo, R.J. La, V. Anantharam, and J.C. Walrand. Analysis and Comparison of TCP Reno and Vegas. In *IEEE INFOCOM (3)*, pages 1556–1563, March 1999.

[54] J. Nagle. Congestion Control in IP/TCP Internetworks. *IETF RFC 896*, January 1984.

[55] M. Neuts. *Queueing Theory: A Linear Algebraic Approach.* MacMillan Publishing Company, New York, 1992.

[56] M. Neuts. *Matrix-Geometric Solutions In Stochastic Modeling: An Algorithmic Approach.* Dover Publications, New York, 1994.

[57] J. Olsén and I. Kaj. Slowstart Window Modeling for Single Connection TCP-Tahoe. In *Seventeenth International Teletraffic Congress (ITC17)*, Salvador da Bahia, Brazil, December 2001.

[58] J. Olsén and I. Kaj. Stochastic Equilibrium Modeling of the TCP Dynamics in Various AQM Environments. Preprint, 2001. Department of Mathematics, Uppsala University.

[59] J. Olsén and I. Kaj. Throughput Modeling and Simulation for Single Connection TCP-Tahoe. In *17th International Teletraffic Congress ITC'01*, Salvador da Bahia, Brazil, December 2001.

[60] T. Ott, J. Kemperman, and M. Mathis. The Stationary Behavior of Ideal TCP Congestion Avoidance. Available from `ftp://ftp.bellcore.com/pub/tjo/TCPwindow.ps`, August 1996.

[61] J. Padhye, V. Firoiu, and D. Towsley. A Stochastic Model of TCP Reno Congestion Avoidance and Control. Technical report, CMPSCI, February 1999.

[62] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP Throughput: A Simple Model and its Empirical Validation. In *ACM SIGCOMM '98*, pages 303–314, Vancouver, CA, September 1998.

[63] C. Partridge and T.J. Shepard. TCP/IP Performance Over Satellite Links. *IEEE Network*, 11(5):44–49, September 1997.

[64] V. Paxson. Automated Packet Trace Analysis of TCP Implementations. In *ACM SIGCOMM '97*, pages 167–179, Cannes, French Riviera, France, September 1997.

[65] V. Paxson. End-to-End Internet Packet Dynamics. In *ACM SIGCOMM '97*, pages 139–152, Cannes, French Riviera, France, September 1997.

[66] V. Paxson and S. Floyd. Wide Area Traffic: the Failure Of Poisson Modeling. *IEEE/ACM Transactions on Networking*, 3(3):226–244, June 1995.

[67] J. B. Postel. Transmission Control Protocol. *IETF RFC 793*, September 1981.

[68] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing.* Cambridge University Press, second edition, 1988-1992.

[69] E. Rosen, A. Viswanathan, and R. Callon. Multiprotocol Label Switching Architecture. *IETF RFC 3031*, January 2001.

[70] M. Roughan, A. Erramilli, and D. Veitch. Network Performance for TCP Networks Part I: Persistent Sources. In *Seventeenth International Teletraffic Congress (ITC17)*, Salvador da Bahia, Brazil, December 2001.

[71] H.-P. Schwefel. *Modeling of Packet Arrivals Using Markov Modulated Poisson Processes with Power-Tail Bursts.* Master's thesis, Institut für Informatik, Technische Universität München, August 1997.

[72] H.-P. Schwefel. *Performance Analysis of Intermediate Systems Serving Aggregated ON/OFF Traffic with Long-Range Dependent Properties.* PhD thesis, Institut für Informatik, Technische Universität München, September 2000.

[73] H.-P. Schwefel. Behavior of TCP-like Elastic Traffic at a Buffered Bottleneck Router. In *IEEE INFOCOM 2001*, volume 3, pages 1698–1705, Anchorage, Alaska, USA, April 2001.

[74] H.-P. Schwefel, M. Jobmann, D. Höllisch, and D. Heyman. On the Accuracy of TCP Performance Models. In *ITCom 2001 Conference on Internet Performance and Control of Network Systems 2*, Denver, Colorado, USA, August 2001.

[75] W. Stevens. *TCP/IP Illustrated, Volume 1.* Addison-Wesley, Reading, Massachusetts, U.S.A., 1994.

[76] W. Stevens. *TCP/IP Illustrated, Volume 2.* Addison-Wesley, Reading, Massachusetts, U.S.A., 1995.

[77] W. Stevens. TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms. *IETF RFC 2001*, January 1997.

[78] M. Villet and A.E. Krzesinski. On the Accuracy of Two TCP Performance Models of MPLS Networks. In *South African Telecommunications, Networks and Applications Conference (SATNAC) '02*, KwaZulu-Natal, South Africa, September 2002.