Abstract

Analytical Characterization of the Throughput of a Split TCP Connection

We develop an analytical characterization of the throughput of a split TCP connection as a function of the packet loss rate in any connection, average round trip time of the connections in series and the receiver's advertised window size. Slowly but steadily, the Internet is moving towards being viewed as a collection of heterogeneous networks by the end parties. In this new scenario, there would be no end-to-end connection between the original sender and the original receiver, instead it will consist of a series of cascaded TCP connections connecting the original sender and receiver. We study the effect on throughput of splitting a TCP connection into two or more connections in series, existing over a wired media. Our work uses the model and notation developed in [1] as a backbone. Our model captures not only the congestion avoidance and fast retransmit mechanisms of TCP, but also includes an analysis of the slow start phase. The number of timeouts were found to increase significantly with an increase in the packet loss rate. The connection in such cases was found to spend considerable time in the slow start stage, thus making the slow start modeling an important part of the analysis. Our analytical characterization of a split TCP connection was validated using the network simulator (ns), version two. The throughput was studied as the packet loss rate and the round trip time were varied over wide ranges. This lead to some insight on the effect of these two network parameters on the throughput of the connection. It was found that for reasonable loss probabilities, a split TCP connection had a much greater throughput than a single TCP connection between the end points.

Author: Ananth Inamti Sundararaj

Advisor: Prof. Dan Duchamp

Date: November 16, 2001

Department: Computer Science

Degree: Master of Science

Acknowledgments

I would like to express my deep sense of gratitude to Prof. Dan Duchamp, Department of Computer Science, Stevens Institute of Technology, Hoboken, for his invaluable and constant guidance and inspiration which enabled me to complete this dissertation. I express my profound thanks to Prof. Duchamp for his keen insight and helpful suggestions. My sincere gratitude to Prof. Stephen Bloom, Director, Department of Computer Science, Stevens Institute of Technology, Hoboken, for granting permission to carry out this work.

Contents

1 Introduction **7**

1.1 Computer Networks and Protocols 7

1.2 Transmission Control Protocol 8

1.2.1 TCP Operation 8

1.2.2 TCP Congestion Control 9

1.2.3 TCP Implementations 10

1.3 Previous Work 10

1.4 Motivation 12

1.5 Contributions 13

1.6 Organization of the Report 15

2 TCP Congestion Control **16**

2.1 Introduction 16

2.2 Split TCP Connection 17

2.3 A Model for TCP Congestion Control 18

3 Congestion Avoidance and Fast Retransmit **23**

3.1 Introduction 23

3.2 Modeling Congestion Avoidance and Fast Retransmit 24

3.2.1 Congestion Control Window 24

3.2.2 Packet Loss Probability 25

3.2.3 Derivation of $E[Y]$ and $E[A]$ 25

3.2.4 Expressions for $E[Y]$ and $E[A]$ 31

4 Timeout and Retransmission **33**

4.1 Introduction 33

4.2 Derivation of Q, $E[R]$ and $E\left[Z^{TO}\right]$ 33

4.2.1 Derivation of Q 33

4.2.2 Derivation of $E[R]$ 35

4.2.3 Derivation of $E\left[Z^{TO}\right]$ 36

4.2.4 Expressions for Q, $E[R]$ and $E\left[Z^{TO}\right]$ 39

5 Slow Start Behavior of TCP **40**

5.1 Introduction 40

5.2 Congestion Control Window Evolution 40

5.2.1 Derivation of $E[L]$ 41

5.2.2 Derivation of $E\left[Z^{SS}\right]$ 44

5.2.3 Expressions for $E[L]$ and $E\left[Z^{SS}\right]$ 45

6 Receiver's Advertised Window Limitation **46**

6.1 Introduction 46

6.2 Effects of Window Limitation 46

6.2.1 Congestion Avoidance 47

6.2.2 Slow Start 50

6.3 Complete Characterization 53

6.3.1 Case I 53

6.3.2 Case II 54

6.3.3 Case III 56

6.3.4 Case IV 57

7 Implementation details **59**

7.1 Role of Simulation 59

7.2 Network Simulator - Version 2 59

7.2.1 Introduction 59

7.2.2 TCP Simulation 60

7.2.3 Ns Limitations 60

7.3 TCP Reno 61

7.4 Simulations 62

7.4.1 Introduction 62

7.4.2 Simulation Configuration 62

8 Results and Discussion **65**

8.1 Introduction 65

8.2 Analysis 65

8.2.1 Comparison Against Ns 65

8.2.2 Comparison with a Single TCP Connection 69

8.3 Conclusions 71

8.4 *N* Split Connections 72

8.5 Future Work 73

A **74**

List of Figures

2.1 A single TCP connection between two points A and C, where A is the sender and C is the receiver 16

2.2 The TCP connection shown in Figure 2.1 being split into $(n+1)$ connections in series, with B_1...B_n being the intermediate senders and receivers 17

2.3 The case of the TCP connection shown in Figure 2.1 being split into two TCP connections, with B being the intermediate receiver and sender 18

2.4 An example of the evolution of the TCP connection 19

8.1 Plot of Duration of Simulation Vs. Agreement 66

8.2 Plot of Round Trip Time Vs. Agreement 67

8.3 Plot of Packet Loss Probability Vs. Agreement 68

8.4 Plot of Packet Loss Probability Vs. Increase 71

Chapter 1

Introduction

1.1 Computer Networks and Protocols

A computer network refers to an interconnected collection of autonomous computers. Two computers are said to be interconnected if they are able to exchange information. By requiring the computers to be autonomous, we wish to exclude from our definition, systems in which there is a clear master/slave relationship [5].

Many different networks exist in the world, often with different software and hardware. There is an obvious need for communication between people connected to these different networks. A collection of such interconnected networks is called an internetwork. To reduce the system complexity, most of the networks are designed as layers. Each layer is responsible for a specific function, and in general, provides some services to the layers above it and makes the implementation details of the lower layers invisible. For example, layer n on one machine connected to a network exchanges information with layer n on another machine connected to another network. The rules and regulations enforced in this information exchange will be the layer n protocol. This protocol will set the ground rules on how communication is to proceed between the communicating agents. As computer networks are developed and deployed in various environments, it is very important to provide reliable means of interconnecting them and to provide standard interprocess communication protocols to support a broad range of applications. *TCP/IP* is the dominant inter-process communication protocol standardization suite in use and is the combination of different protocols at various layers.

TCP/IP is considered to be a five layer system: the physical layer, the link layer, the network layer, the transport layer and the application layer. The transport layer

can be looked upon as the heart of the whole protocol hierarchy. It provides data transport for the application layer above it. *Transmission Control Protocol* (TCP) and *User Datagram Protocol* (UDP) are two different transport protocols in the TCP/IP protocol suite. The transport protocol used in a particular situation depends on the concerned application. Our work deals with the transmission control protocol (TCP).

1.2 Transmission Control Protocol

TCP is a connection-oriented, end-to-end protocol designed to provide for reliable inter-process communication between pairs of processes. Its functions, amongst others, involve fragmenting data received from the application layer into appropriate size for the network layer below it, acknowledging received packets, etc. Since the transport layer provides a reliable flow of data, the application layer above it can ignore all such details. TCP operates above a wide range of communication systems ranging from hardwired connections to satellite networks [7].

1.2.1 TCP Operation

The primary purpose of the TCP is to provide reliable connection service between pairs of processes [18]. TCP's protocol specification is provided in [18]. To allow for many processes within a single host to use TCP communication facilities simultaneously, the TCP provides a set of addresses or ports within each host. Concatenated with the network and host addresses from the Internet communication layer, this forms a socket. A pair of sockets uniquely identify each connection. That is, a socket may be simultaneously used in multiple connections.

TCPs initialize and maintain certain status information for each data stream. The combination of this information, including sockets, sequence numbers, and window sizes, is called a connection. Each connection is uniquely specified by a pair of sockets identifying its two sides. When two processes wish to communicate, their TCP's must first establish a connection (initialize the status information on each side). When their communication is complete, the connection is terminated or closed to free the resources for other uses.

TCP is able to transfer a continuous stream of octets in each direction between its users by packaging some number of octets into segments for transmission on the connection. In general, the TCP's decide when to block and forward data at their own convenience. The TCP must recover from data that is damaged, lost,

duplicated, or delivered out of order by the Internet communication system. This is achieved by assigning a sequence number to each octet transmitted, and requiring a positive acknowledgment (ACK) from the receiving TCP. If the ACK is not received within a timeout interval, the data is retransmitted. At the receiver, the sequence numbers are used to correctly order segments that may be received out of order and to eliminate duplicates. Damage is handled by adding a checksum to each segment transmitted, checking it at the receiver, and discarding damaged segments. TCP provides a means for the receiver to govern the amount of data sent by the sender. This is achieved by returning a *window* with every ACK, indicating a range of acceptable sequence numbers beyond the last segment successfully received. The window indicates an allowed number of octets that the sender may transmit before receiving further permission.

1.2.2 TCP Congestion Control

TCP congestion control is described in RFC 2581 [19]. TCP uses a flow control based on windows [32]. A source keeps a variable called *window size* that determines the maximum number of outstanding packets that have been transmitted but not yet acknowledged. A connection starts with a small window size of one packet and the source increments its window by one each time it receives an acknowledgement. This doubles the window once every round trip time. This is implemented in the slow start algorithm. When the window reaches a threshold value, the connection enters the congestion avoidance phase, where on the receipt of each acknowledgement the window is increased by the reciprocal of its current size. The threshold determines the transition from slow start to congestion avoidance and is meant to indicate the available network capacity and is adjusted each time a loss is detected. On detecting a loss, the slow start threshold is set to half of the current window size, the lost packet is retransmitted and the connection re-enters slow start by resetting its window to one. Fast recovery and fast retransmit algorithms are used to recover from a loss more efficiently. If three or more duplicate acknowledgments are received in a row, it is a strong indication that a packet has been lost. Hence the missing packet is retransmitted without waiting for the retransmission timer to expire. This is the fast retransmit algorithm. Next, instead of performing slow start, congestion avoidance is performed, as the receipt of acknowledgments suggests that packets have left the network. This is the fast recovery algorithm.

1.2.3 TCP Implementations

The original definition of TCP appears in RFC 793 [18]. RFC 1122 [34] updates and supplements the definition. To meet the TCP standard, an implementation must follow RFC 793 and RFC 1122. Although these two documents give a detailed description of TCP implementation, two TCP implementations that conform to the specifications can differ slightly because an implementor has some freedom to choose a software design, parameters, and to interpret the protocol standards [33].

The predominant TCP implementations are called Tahoe and Reno. The basic idea of the Tahoe version of TCP is to gently probe the network for spare capacity by linearly increasing its rate and by exponentially reducing its rate when congestion is detected by a packet loss. It implements the slow start and congestion avoidance algorithms. TCP Reno included the implementation of the fast retransmit and fast recovery algorithms to recover from losses more efficiently [32].

TCP Vegas improves upon TCP Reno through three main techniques. The first is a new retransmission mechanism where timeout is checked on receiving the first duplicate acknowledgement, rather than waiting for the third duplicate acknowledgement, as TCP Reno would. This results in a more timely detection of loss. The second technique is a more prudent way to grow the window size during the initial use of slow start, when a connection starts up, and it results in fewer losses. The third technique is a new congestion avoidance mechanism that corrects the oscillatory behavior of TCP Reno.

1.3 Previous Work

Over the past few years a lot of attention has been paid to TCP modeling within the research community [1,2]. This might be due to the fact that more than ninety percent of today's Internet traffic is carried by TCP [2]. Closed form expressions for TCP throughput have been obtained. These expressions have aided in understanding the effect of various network and TCP parameters on the throughput of the connection and on the efficiency of network resource utilization. These expressions have been used to implement TCP friendly flow control for unicast applications that do not utilize TCP at the transport layer, but compete for bandwidth with a TCP connection [13]. The need for all applications to implement some form of congestion control is increased due to the fact that applications that largely ignore congestion issues could lead to widespread congestive collapse in the Internet [7]. To implement an equivalent to TCP's congestion control, it is imperative that the

effect of the packet loss rate on the overall performance of a TCP connection be understood.

A simple model of the TCP Reno protocol has been proposed in [1]. A simple analytic characterization of the steady state throughput, as a function of loss rate, round trip time (RTT) and the receiver's advertised window, for a bulk transfer TCP flow has been developed. The model captures the essence of TCP's congestion avoidance behavior and expresses throughput as a function of loss rate, RTT and the receiver's advertised window. The model takes into account the behavior of the protocol in the presence of time outs, and is valid over the entire range of loss probabilities. It was observed that most of the real world connections suffered from a significant number of timeouts. A simplified expression for the TCP bandwidth was presented and was found to be a good approximation for the proposed model in most cases. However, it has been assumed that the time spent in slow start is negligible and hence the slow start algorithm has not been modeled. Further, an average, based on observed tcpdump data, has been assumed for the number of packets sent between two timeouts in sequence for the same unacknowledged packets.

An analysis of TCP throughput under a general loss process has been presented in [2]. The only assumptions made on the loss process under consideration were those of stationarity and ergodicity. An explicit expression for the throughput in the case of no limit on the transmission rate was provided. The throughput was shown to be inversely proportional to RTT and to the square root of the packet loss probability. Further, bounds for the case when a limit exists on the maximum window size were also provided. The work was extended to include the timeout mechanism as well.

Many TCP applications are based on the transfer of small files. Hence the entire transfer may occur within the slow start phase. In that case the TCP connection is not able to utilize all available resources in the network [7]. Several solutions have been put forth to cope with performance problems encountered in TCP.

One such solution is TCP spoofing[8]. In this, a router near the source sends back the ACKs for TCP packets in order to give the source the illusion of a short delay path. TCP spoofing has been found to improve throughput, but suffers from the fact that spoofing requires ACKs to flow through the same path as the data. On the contrary, in the Internet, it is possible for ACKs to flow through a different path than data. To overcome this defect a split or cascading TCP connection was proposed[3,8].

A split connection protocol will split the TCP connection between the sender and the receiver into two, or possibly more, connections at intermediate points. Previous works on Split TCP connections have studied connections which spread over

different communication mediums[3,4,8]. They have concentrated on splitting the connection at the boundary of separate communication mediums so as to isolate issues pertaining to the different mediums. The design and implementation of Indirect-TCP, an indirect transport layer protocol for mobile hosts, has been described in [3,4]. In this, an end-to-end TCP connection between a fixed host and a mobile host is split into two separate connections. One being the regular TCP connection between the fixed host and the base station currently serving the mobile host and the other a wireless TCP connection between the mobility support router and the mobile host. The basic idea being that in cases involving interconnection between hosts on the internetwork, such as between a mobile host and a stationary host, communication takes place over two drastically different kinds of mediums (e.g., wireless and wired). These connections are then split into two separate interactions, one for each kind of communication medium. Thus isolating mobility and wireless related problems using the base stations as intermediaries, which also provide backward compatabilty with fixed network protocols. The work attempts to separate the loss recovery over the wireless links from that across the wireline network, thereby shielding the original TCP sender from the wireless link.

Recently, there has been a great deal of activity in the area of transparent proxies for web caching [25]. Transparent proxies are commonly used in solutions when an application is to be proxied in a manner that is completely oblivious to a client, without requiring any prior configuration. To ensure that all IP packets of an intercepted TCP connection are seen by the intercepting transparent proxy, they must sit at focal points in the network. This limitation has been overcome in [25] by using TCP options and IP tunneling to ensure that all IP packets belonging to a TCP connection will traverse the proxy that intercepted the first TCP packet. Translucent Proxying of TCP (TPOT) protocol is discussed in [25]. It describes a prototype implementation and analyzes its impact on the performance of TCP.

1.4 Motivation

We attempt to study the effect on throughput of splitting a TCP connection into two or more connections in series, existing over a wired media. As noted above, previous work on split connections have all dealt with connections extending over different communication mediums, namely wired and wireless. The basic idea there being to isolate and independently deal with issues arising out of the use of different mediums in a single connection. It has already been found that by shielding the original TCP sender from the wireless link the throughput can be increased.

The Internet as it exists today, is visible as one big homogeneous network with no distinct boundaries to the end parties of a TCP connection. This is due to the end-to-end semantics of the transport layer. Slowly but steadily, the Internet is moving towards being viewed as a collection of heterogeneous networks by the end parties. In this new scenario, there would be no end-to-end connection between the original sender and the original receiver, instead it will consist of a series of cascaded TCP connections connecting the original sender and receiver. The intermediate connections might be across some often traversed points, thereby giving enough reason for them to be connected for an extended duration, rather than the connection between them being set up and torn down ever so often. Thus reducing the connection establishment/tear down overhead. But before more work is done on the same, three issues need to be investigated further:

1. What are the exact mechanics of carrying out this partition of the Internet, as present today, into a heterogenous collection of networks with distinct boundaries
2. What are the overheads, positives and negatives, involved in this new arrangement of the Internet, and issues pertaining to the overhead in splitting a connection
3. What would be the effect on performance, namely on throughput, of the splitting of the connection

If all the three questions were to have favorable answers then revolutionary work on rearranging the Internet could be attempted. In this work we propose to answer the third question: the effect on throughput, performance in general, of splitting a TCP connection. We found the answer to this question to be favorable.

1.5 Contributions

We used the model developed in [1] as a backbone for our analysis. The contributions of this work include:

1. Developing an expression for the throughput of a split TCP connection as a function of the packet loss rate in any connection, average RTT of the connections in series and the receiver's advertised window size
2. Modeling the slow start intervals in addition to modeling the congestion avoidance and fast retransmit mechanisms

3. Considering the average of the number of packets sent between two timeouts in sequence for the same unacknowledged packets to be an unestablished constant *d*. The analysis in [1] assumes this constant to be one, based on observed tcpdump data

4. Correcting an error in the modeling of timeouts as described in [1]. $Q\prime(w)$, the probability that a loss in a window of size w is a timeout was described to be

$$Q\prime(w) = \begin{cases} 1 & w \leq 3 \\ \sum_{k=0}^{2} A(w,k) + \left(\sum_{k=3}^{w} A(w,k) * \sum_{m=0}^{2} C(k,m)\right) & otherwise \end{cases}$$

where $A(w,k)$ denotes the probability that the first k packets are acknowledged in a round of w packets sent in the penultimate round and $C(n,m)$ is the probability that m packets are acknowledged in sequence in the last round, where n packets are sent. But when $w > 3$, $Q\prime(w)$ should be defined as

$$Q\prime(w) = \sum_{k=0}^{2} A(w,k) + \left(\sum_{k=3}^{w-1} A(w,k) * \sum_{m=0}^{2} C(k,m)\right)$$

The second summation, $\sum_{k=3}^{w} A(w,k)$, in the expression should be $\sum_{k=3}^{w-1} A(w,k)$. Else, when $k = w$, we would have $A(w,w)$ i.e. all w packets are successfully sent in the penultimate round, but the round would be the penultimate round if and only if a loss occurs in this round. Hence the summation has to be

$$A(w,3) + A(w,4) + ... + A(w,w-1)$$

or

$$\sum_{k=3}^{w-1} A(w,k)$$

5. Analyzing the effect of the receiver's advertised window on the throughput of the split connection while the connection is in congestion avoidance and also while in the slow start phase

Our analytical characterization of a split TCP connection was validated using the network simulator (ns), version two. The throughput was studied as the packet loss rate and RTT were varied over wide ranges. This lead to some insight on the effect of the these two network parameters, namely packet loss rate and RTT, on the throughput of the connection. It was found that for reasonable loss probabilities, a split TCP connection had a much greater throughput than a single TCP connection between the end points.

1.6 Organization of the Report

The rest of the report is organized as follows:

Chapter 2 proposes a model for the TCP congestion control behavior

Chapter 3 develops a model for the congestion avoidance and fast retransmit algorithms

Chapter 4 models the behavior of TCP on the timeout of the retransmission timer. In addition, subtle issues such as the number of packets sent between two timeouts in sequence for the same unacknowledged packets have also been discussed

Chapter 5 develops expressions for the number of packets sent in the slow start stage and the duration of time the connection spends in slow start

Chapter 6 studies the effect of the receiver's advertised window on the throughput of the split connection and puts together all the previously developed expressions to obtain a final expression for TCP throughput

Chapter 7 gives the implementation details

Chapter 8 presents a discussion of the results and some suggestions for future work

Chapter 2

TCP Congestion Control

2.1 Introduction

We carried out an analytical characterization of a split TCP connection and obtained an expression for the throughput of the whole connection as a function of the packet loss probability in any link and the round trip time of the connections in series. It should however be noted that when we use the word throughput, we do not refer to the good-put of the connection. We are interested in the total packets sent in the connection irrespective of their eventual fate. But it should be noted that it is already established that these packets have successfully traversed all but the last cascaded connection. They have been transmitted on the last cascaded connection and have a certain probability of reaching the destination, the end receiver.

We use the model as developed in [1] as a backbone, we modify it to a split TCP connection scenario and also include analysis of the slow start stage. Most of the notation and analysis used here is the same as that developed in [1]. But for the sake of completeness we will start the modeling and development of all expressions from first principles.

Consider a TCP connection from A to C, as shown below

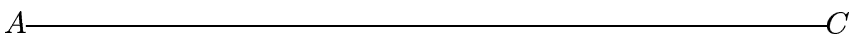

Figure 2.1: A single TCP connection between two points A and C, where A is the sender and C is the receiver

The following expression for TCP throughput, $B(p)$, was developed in [1]

$$B(p) = \begin{cases} \frac{\frac{1-p}{p} + E[W_u] + Q\prime(E[W_u])\frac{1}{1-p}}{RTT\left(\frac{b}{2}E[W_u]+1\right)+Q\prime(E[W_u])T_o\frac{f(p)}{1-p}} & if\ E[W_u] < W_{max} \\ \frac{\frac{1-p}{p} + W_{max} + Q\prime(W_{max})\frac{1}{1-p}}{RTT\left(\frac{b}{8}W_{max}+\frac{1-p}{pW_{max}}+2\right)+Q\prime(W_{max})T_o\frac{f(p)}{1-p}} & otherwise \end{cases} \tag{2.1}$$

where $B(p)$ is the TCP throughput

p is the probability that a packet is lost, given that either it is the first packet in it's round or the preceding packet in it's round is not lost

$f(p)$ is defined as

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

$E[W_u]$ is the expected value of the unconstrained congestion control window size

W_{max} is the receiver advertised maximum buffer size

$Q\prime(w)$ is the probability that a loss in a window of size w is a timeout

T_o is the period of time that a sender waits before retransmitting non-acknowledged packets

b is the number of packets acknowledged by each ACK

2.2 Split TCP Connection

The TCP connection between A and C, as shown in figure 2.1, may be split at n places to yield $(n+1)$ TCP connections in series between A and C as shown next

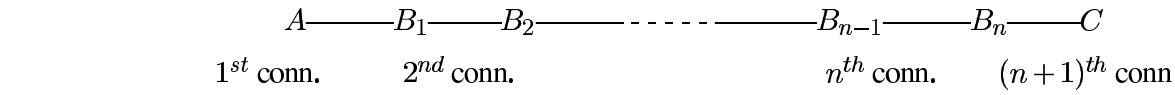

Figure 2.2: The TCP connection shown in Figure 2.1 being split into $(n+1)$ connections in series, with B_1...B_n being the intermediate senders and receivers

Now our objective is to model this scenario and develop an expression for the throughput of this split TCP connection as a function of the packet loss rate in any connection, the average round trip time of the TCP connections in series and the

receiver's advertised window size. For the sake of clarity and simplicity, instead of developing a model for $(n+1)$ connections in series, we will take the simplest and most likely case, consisting of two connections in series. So Figure 2.2 simplifies to Figure 2.3 as shown below.

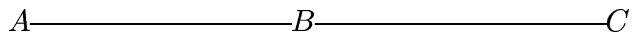

Figure 2.3: The case of the TCP connection shown in Figure 2.1 being split into two TCP connections, with B being the intermediate receiver and sender

Now these two TCP connections in series would be observed for a certain period of time. The throughput for the observed interval of time would be the number of packets sent in the observed interval (say some value X) divided by the duration of the observed interval (say some value Z). So we would have $Throughput = \frac{X}{Z}$. The numerator, namely the number of packets sent in the observed interval, would be the number of packets received successfully by the intermediate point B and, after some processing perhaps, sent by the second TCP connection, plus the number of packets still in transit in the first TCP connection, which have a certain probability of reaching their destination. When the number of TCP connections in series is small (two, three) (six, seven would be considered large) the former number will dominate the total count, such that we can ignore the packets in transit in the first and following connections except of course the last connection. So now onwards all discussion will be based on the two connections in series case as shown in Figure 2.3 above. Further, we will always be referring to the second TCP connection unless explicitly stated otherwise.

2.3 A Model for TCP Congestion Control

We will develop a probabilistic model for the TCP congestion control mechanism. A sequence number uniquely identifies a packet in the stream of data from the sending TCP to the receiving TCP. A packet which acknowledges data received by the receiving TCP contains an acknowledgement number, which is the next sequence number that the sender of the acknowledgement expects to receive.

TCP has four intertwined congestion control algorithms: slow start, congestion avoidance, fast retransmit and fast recovery. Beginning transmission into a network with unknown conditions requires TCP to slowly probe the network to determine the available capacity, in order to avoid congesting the network with an inappropriately large burst of data. The slow start algorithm is used for this purpose at the beginning of the transfer. Two variables, *cwnd* and *ssthresh,* are used to

implement the algorithms. The congestion control window (cwnd) is a sender-side limit on the amount of data the sender can transmit into a network before receiving an ACK. The state variable, slow start threshold (ssthresh), is used to determine whether the slow start or congestion avoidance algorithm is used to control data transmission. Initially TCP sets cwnd to one unit and ssthresh to an arbitrarily high value, typically 65535 bytes. In our discussion one unit is a packet. During slow start, TCP increments cwnd by one for each ACK received that acknowledges new data. Note that when sending data, TCP sends the minimum of the receiver's advertised window and cwnd. Slow start ends when cwnd exceeds ssthresh or when congestion is observed. If the cwnd exceeds ssthresh then congestion avoidance takes over. When the connection is in congestion avoidance, on the receipt of every non-duplicate ACK, cwnd is increased by $\left(\frac{1}{cwnd}\right)$.

The presence of congestion in the network might be indicated in two ways: expiry of a retransmission timer or the receipt of a triple duplicate acknowledgement. When a packet is transmitted a retransmission timer is started, if the timer expires before an ACK acknowledging that packet is received, a timeout is said to have occurred. On the occurrence of a timeout the TCP sets cwnd to one, retransmits the non-acknowledged packet and then goes into slow start. Congestion may also be indicated by the receipt of a triple duplicate ACK. Then the fast retransmit and fast recovery algorithm take over. On the receipt of a triple duplicate ACK, the ssthresh is set to one half of the current window size. The non-acknowledged packet is then retransmitted and cwnd is set to three plus the value of ssthresh. For each additional duplicate ACK received, cwnd is incremented by one. This artificially inflates the congestion window in order to reflect the fact that additional packets have left the network. If allowed by the new value of cwnd and the receiver's advertised window, a packet is transmitted. When the next ACK arrives that acknowledges new data, cwnd is set to ssthresh. This is termed *deflating* the window, as cwnd has been set to one half its value when congestion was indicated. Now the connection is in congestion avoidance and the window growth and packet transmission proceed as described above.

An example of the evolution of the TCP connection could be

slow start.....congestion avoidance.....time out.....slow start.....triple duplicate loss indication.....fast retransmit.....congestion avoidance.....triple duplicate loss indication.....fast retransmit.....congestion avoidance.....time out.....slow start.....

Figure 2.4: An example of the evolution of the TCP connection

Let

Z_i^{SS} denote the duration when the connection is in slow start

Z_i^{TO} denote the duration of a sequence of time outs

Z_i^{TD} denote the duration of a sequence of triple duplicate periods, the last of which ends in a time out.

A Triple Duplicate Period (TDP) would be defined to be the period between two Triple Duplicate (TD) loss indications, or between a TD loss indication and a time-out loss indication without any other loss indications in between.

Define

$$S_i = Z_i^{SS} + Z_i^{TO} + Z_i^{TD} \tag{2.2}$$

M_i to be the number of packets sent during S_i

If $(S_i, M_i)_i$is an independent identically distributed sequence of random variables, then the throughput of the TCP connection [14], B, is defined as

$$B = \frac{E[M]}{E[S]} \tag{2.3}$$

Let

n_i be the number of TD periods in Z_i^{TD}

For the j^{th} TD period of the interval Z_i^{TD}, we define

Y_{ij} to be the number of packets sent in the period

A_{ij} to be the duration of the period

X_{ij} to be the number of rounds in the period

W_{ij} to be the window size at the end of the period

Further, let

R_i denote the total number of packets transmitted in Z_i^{TO}

L_i denote the number of packets sent in Z_i^{SS}

Then we have

$$M_i = \sum_{j=1}^{n_i} Y_{ij} + R_i + L_i \tag{2.4}$$

$$S_i = \sum_{j=1}^{n_i} A_{ij} + Z_i^{TO} + Z_i^{SS} \tag{2.5}$$

Thus we have

$$E[M] = E\left[\sum_{j=1}^{n_i} Y_{ij}\right] + E[R] + E[L] \tag{2.6}$$

and

$$E[S] = E\left[\sum_{j=1}^{n_i} A_{ij}\right] + E\left[Z^{TO}\right] + E\left[Z^{SS}\right] \tag{2.7}$$

Now, if $\{n_i\}_i$ is an independent identically distributed sequence of random variables, independent of $\{Y_{ij}\}$ and $\{A_{ij}\}$ then

$$E\left[\left(\sum_{j=1}^{n_i} Y_{ij}\right)_i\right] = E[n]E[Y] \tag{2.8}$$

and

$$E\left[\left(\sum_{j=1}^{n_i} A_{ij}\right)_i\right] = E[n]E[A] \tag{2.9}$$

Now to arrive at $E[n]$, we note that in Z_i^{TD} there are n TDPs and the first $(n-1)$ TDPs end in a TD and the last TDP ends in a timeout. So we have that in a Z_i^{TD}, there is one timeout loss indication out of n_i loss indications. Let Q represent the probability that the loss indication ending a TDP is a timeout loss indication, so

$$Q = \frac{1}{E[n]} \tag{2.10}$$

Using (2.3), (2.6), (2.7), (2.8) and (2.9) we have

$$B = \frac{E[n]E[Y] + E[R] + E[L]}{E[n]E[A] + E\left[Z^{TO}\right] + E\left[Z^{SS}\right]}$$

and using (2.10) we have

$$B = \frac{E[Y] + Q * (E[R] + E[L])}{E[A] + Q * (E\left[Z^{TO}\right] + E\left[Z^{SS}\right])} \tag{2.11}$$

Most of the analysis carried out above was developed in [1]. It is being reproduced here for convenience. We will next obtain the expressions for the variables in the above equation.

Chapter 3

Congestion Avoidance and Fast Retransmit

3.1 Introduction

Congestion avoidance is a way to deal with lost packets. A packet loss is assumed to be an indication of the presence of congestion in the network rather than an indication of a link failure. Hence, when a packet loss occurs, we wish to slow down the transmission rate of the packets into the network. We do this by reducing the congestion control window size and retransmitting the unacknowledged packets. When new data is acknowledged by the other end we increase the congestion control window size. The way it increases depends on whether the packet loss was indicated by a timeout or by the receipt of a triple duplicate acknowledgement. In other words, it depends on whether we are performing slow start or congestion avoidance.

We model the congestion avoidance and fast retransmit algorithms of TCP taking into account the fact that a packet loss indication is a triple duplicate acknowledgement rather than the expiry of a retransmission timer. Since Y_{ij} and A_{ij} do not depend on time outs, we will obtain the expressions for $E[Y]$ and $E[A]$. That is, the expected value for the number of packets sent while the connection was in congestion avoidance and the duration of time the TCP connection spends in congestion avoidance, respectively.

3.2 Modeling Congestion Avoidance and Fast Retransmit

3.2.1 Congestion Control Window

Let W denote TCP's congestion control window size. Now since we assume that we are in congestion avoidance, it implies that W will be incremented by $\frac{1}{W}$ each time an ACK is received. Further, whenever packet loss is indicated by the receipt of a triple duplicate (TD) ACK, the window size would be decremented. The congestion avoidance mechanism is modeled in terms of rounds. A round starts with the back to back transmission of W packets, where W is the current congestion control window size. Once all the packets falling within the congestion control window have been sent, no other packet is sent until the first ACK is received for one of these W packets. The receipt of this ACK marks the end of the current round and the beginning of the next round. The later acknowledgments of the previous round will be received in the current round and could be considered as coinciding with the transmission of the packets in the current round. The duration of the round is the same as the round trip time (RTT) and is assumed to be independent of the window size. Further, it is assumed that the time required to send W packets is smaller than the round trip time. At the beginning of the next round, $W\prime$ new packets will be sent, where $W\prime$ would be the new size of the congestion control window, b is the number of packets that are acknowledged by a received ACK. Each ACK increases the window size by $\frac{1}{W}$, hence the window size at the beginning of the next round is

$$W' = W + \left(\frac{1}{W} * \frac{W}{b}\right)$$

or

$$W' = W + \frac{1}{b} \tag{3.1}$$

In other words, during congestion avoidance and in the absence of loss, the window size increases linearly in time, with a slope of $\frac{1}{b}$ packets per round trip time. The assumptions made are:

1. A packet is lost in a round independently of any other packet lost in any other round

2. If a packet is lost then all remaining packets transmitted until the end of the round are lost as well

3.2.2 Packet Loss Probability

We define p to be the probability that a packet is lost in a connection, given that it is the first packet in its round or that the preceding packet in its round is not lost. The probability that a packet gets lost in any round in the second connection depends on its probability of getting lost in the first connection, which is p. The probability that a packet does not get lost in the first connection is $(1-p)$. For a packet to be transmitted in the second connection, it has to be successfully received by the intermediate point. Then as noted earlier, that will be followed by some processing and then it is transmitted on the second connection. We make an assumption that no packets are lost during this processing at the intermediate point.

So now the probability that a packet does not get lost in the second round would be $(1-p) * (1-p)$ or $(1-p)^2$

So the probability that a packet does get lost in the second connection is $\left(1-(1-p)^2\right)$

3.2.3 Derivation of $E[Y]$ and $E[A]$

A TD period starts immediately after a TD loss indication. Considering the i^{th} TD period :

the current congestion control window is $\frac{W_{i-1}}{2}$

We denote by α_i the first packet lost in the i^{th} TDP, i.e. until such time and including it, α_i packets have been transmitted

X_i denotes the round where this loss occurred.

After packet α_i , a maximum of $W_i - 1$ more packets might be sent in an additional round before the current TD period ends. This will be the case when the last packet sent in the penultimate round is the first packet lost. So for every acknowledged packet, which was sent in the penultimate round, the window size would be increased and a packet would be transmitted in the final round, if permitted.Thus a maximum of $Y_i = \alpha_i + W_i - 1$ packets are sent in $X_i + 1$ rounds.

It follows that

$$E[Y] = E[\alpha] + E[W] - 1 \tag{3.2}$$

Derivation of $E[\alpha]$

Next to derive $E[\alpha]$, we consider the random process $\{\alpha_i\}_i$, which is a sequence of independent and identically distributed random variables, since the packets are

lost independently of any other packets lost in any other rounds. The probability that $\alpha_i = k$ is equal to the probability that exactly $(k-1)$ packets are successfully acknowledged before a loss occurs, hence

$$P[\alpha = k] = (1-p)^{2(k-1)} \left(1 - (1-p)^2\right), \; k = 1, 2, 3 \ldots \tag{3.3}$$

The mean of α is

$$E[\alpha] = \sum_{k=1}^{\infty} (1-p)^{2(k-1)} \left(1 - (1-p)^2\right) k \tag{3.4}$$

or

$$E[\alpha] = \left(1 - (1-p)^2\right) \sum_{k=1}^{\infty} (1-p)^{2(k-1)} k$$

or

$$E[\alpha] = \left(1 - (1-p)^2\right) \left[1 + 2(1-p)^2 + 3(1-p)^4 + 4(1-p)^6 + ...\right]$$

or

$$E[\alpha] = \left(1 - (1-p)^2\right) \left[\frac{1}{\left(1 - (1-p)^2\right)} + \frac{(1-p)^2}{\left(1 - (1-p)^2\right)^2}\right]$$

or

$$E[\alpha] = \left(1 - (1-p)^2\right) \left[\frac{\left(1 - (1-p)^2\right) + (1-p)^2}{\left(1 - (1-p)^2\right)^2}\right]$$

or

$$E[\alpha] = \frac{1}{\left(1 - (1-p)^2\right)} \tag{3.5}$$

Substituting (3.5) in (3.2) we have

$$E[Y] = \frac{1}{\left(1 - (1-p)^2\right)} - 1 + E[W]$$

or

$$E[Y] = \frac{1 - \left(1 - (1-p)^2\right)}{\left(1 - (1-p)^2\right)} + E[W]$$

or

$$E[Y] = \frac{(1-p)^2}{\left(1 - (1-p)^2\right)} + E[W] \tag{3.6}$$

Derivation of $E[W]$

We next describe the derivation of $E[W]$

We note that in the i^{th} TD period, the window size increases from $\frac{W_{i-1}}{2}$ to W_i .

In other words

at the beginning of the first round the window size is $\frac{W_{i-1}}{2}$

at the beginning of the second round it is $\frac{W_{i-1}}{2} + \frac{1}{b}$

at the beginning of the third round the window size is $\frac{W_{i-1}}{2} + \frac{2}{b}$, and so on.

The window size increases linearly with a slope of $\frac{1}{b}$. So at the beginning of the last round the window size would be W_i

where

$$W_i = \frac{W_{i-1}}{2} + \frac{X_i}{b},\ i = 1, 2, 3 \ldots \tag{3.7}$$

Note the assumption that $\frac{W_{i-1}}{2}$and $\frac{X_i}{b}$ are integers.

Since in TDP_i , Y_i packets are transmitted, we see that $\frac{W_{i-1}}{2}$ packets are sent in each of the first b rounds, then $\left(\frac{W_{i-1}}{2} + 1\right)$ are sent in each of the next b rounds. So the total number of packets sent would be

$$Y_i = \left[\underbrace{\left(\frac{W_{i-1}}{2} + \ldots \frac{W_{i-1}}{2} \right)}_{\text{b times}} + \ldots + \underbrace{\left(\left(\frac{W_{i-1}}{2} + \left(\frac{X_i}{b} - 1 \right) \right) + \ldots + \left(\frac{W_{i-1}}{2} + \left(\frac{X_i}{b} - 1 \right) \right) \right)}_{\text{b times}} \right] + \beta_i \tag{3.8}$$

where β_i is the number of packets sent in the last round. We now have

$$Y_i = b\left(\frac{W_{i-1}}{2}\right) + b\left(\frac{W_{i-1}}{2}+1\right) + ... + b\left(\frac{W_{i-1}}{2} + \left(\frac{X_i}{b}-1\right)\right) + \beta_i$$

or

$$Y_i = b\left[\left(\frac{W_{i-1}}{2}\right) + \left(\frac{W_{i-1}}{2}+1\right) + ... + \left(\frac{W_{i-1}}{2} + \left(\frac{X_i}{b}-1\right)\right)\right] + \beta_i$$

or

$$Y_i = b\left[\sum_{k=0}^{\frac{X_i}{b}-1}\left(\frac{W_{i-1}}{2}+k\right)\right] + \beta_i \tag{3.9}$$

or

$$Y_i = b\left[\frac{X_i}{b}\left(\frac{W_{i-1}}{2}\right) + 1 + 2 + 3 + ... + \left(\frac{X_i}{b}-1\right)\right] + \beta_i$$

or

$$Y_i = b\left[\left(\left(\frac{W_{i-1}}{2}\right) * \frac{X_i}{b}\right) + \left(\left(\frac{X_i}{b}-1\right) * \frac{X_i}{b} * \frac{1}{2}\right)\right] + \beta_i$$

or

$$Y_i = \frac{X_i}{2} * \left[W_{i-1} + \left(\frac{X_i}{b}-1\right)\right] + \beta_i \tag{3.10}$$

Substituting (3.7) in (3.10) we have

$$Y_i = \frac{X_i}{2} * \left[W_i + \frac{W_{i-1}}{2} - 1\right] + \beta_i \tag{3.11}$$

Next, if we assume that $\{X_i\}$ and $\{W_i\}$ to be mutually independent sequences of independent identically distributed random variables, we have

$$E[W] = \frac{2}{b} * E[X] \tag{3.12}$$

and

$$E[Y] = \frac{E[X]}{2}\left(\frac{E[W]}{2} + E[W] - 1\right) + E[\beta] \tag{3.13}$$

As β_i can be assumed to be uniformly distributed between 0 and W_i , we have

$$E[\beta] = \frac{E[W]}{2} \tag{3.14}$$

Substituting (3.12) and (3.14) in (3.13) we have

$$E[Y] = \frac{b}{4}E[W]\left(\frac{E[W]}{2} + E[W] - 1\right) + \frac{E[W]}{2} \tag{3.15}$$

Substituting (3.6) in (3.15) we have

$$\frac{(1-p)^2}{\left(1-(1-p)^2\right)} + E[W] = \frac{b}{4}E[W]\left(\frac{E[W]}{2} + E[W] - 1\right) + \frac{E[W]}{2} \tag{3.16}$$

Let $\frac{(1-p)^2}{\left(1-(1-p)^2\right)}$ be C, then the above equation can be written as

$$C + E[W] = \frac{b}{4}E[W]\left(\frac{E[W]}{2} + E[W] - 1\right) + \frac{E[W]}{2}$$

or

$$C + E[W] = \frac{b}{4}E[W]\left(\frac{3}{2}E[W] - 1\right) + \frac{E[W]}{2}$$

or

$$E[W]^2\left(\frac{3b}{8}\right) + E[W]\left(\frac{-1}{2} + \frac{-b}{4}\right) - C = 0 \tag{3.17}$$

The equation, (3.17), is a quadratic equation of the form $ax^2 + bx + c = 0$ where

$$x = E[W]$$

$$a = \frac{3b}{8}$$

$$b = \left(\frac{-1}{2} + \frac{-b}{4}\right)$$

$$c = -\frac{(1-p)^2}{\left(1-(1-p)^2\right)}$$

Hence we have

$$E[W] = \frac{\left(\frac{1}{2}+\frac{b}{4}\right) + \sqrt{\left(\frac{1}{2}+\frac{b}{4}\right)^2 + 4 * \frac{3b}{8}\left(\frac{(1-p)^2}{\left(1-(1-p)^2\right)}\right)}}{\frac{3b}{4}}$$

or

$$E[W] = \frac{(2+b)}{3b} + \sqrt{\left(\frac{b+2}{3b}\right)^2 + \frac{8}{3b}\left(\frac{(1-p)^2}{(1-(1-p)^2)}\right)} \tag{3.18}$$

Expression for $E[Y]$

Substituting (3.18) in (3.6) we have

$$E[Y] = \frac{(1-p)^2}{\left(1-(1-p)^2\right)} + \frac{(2+b)}{3b} + \sqrt{\left(\frac{b+2}{3b}\right)^2 + \frac{8}{3b}\left(\frac{(1-p)^2}{\left(1-(1-p)^2\right)}\right)} \tag{3.19}$$

Expression for $E[X]$

From (3.12) and (3.18) we have

$$E[X] = \frac{(2+b)}{6} + \sqrt{\left(\frac{b+2}{6}\right)^2 + \frac{2b}{3}\left(\frac{(1-p)^2}{\left(1-(1-p)^2\right)}\right)} \tag{3.20}$$

Derivation of $E[A]$

Next to derive $E[A]$, consider the i^{th} TDP, let r_{ij} be the duration of the j^{th} round of TDP_i . Hence the duration of TDP_i is A_i

$$A_i = \sum_{j=1}^{X_i+1} r_{ij} \tag{3.21}$$

Since r_{ij} are considered to be random variables, assumed to be independent of the size of the congestion window and hence independent of the round number j. Further, it has been assumed that the time required to send all the packets in a round is smaller than the round trip time.

It follows that

$$E[A] = (E[X] + 1)\, E[r] \tag{3.22}$$

$E[r]$ is denoted by RTT, the average value of the round trip time for the connections in series under consideration, which in our case is the second connection of the split TCP connection. We again note the point that, for the sake of simplicity, we assume that the RTT of the second connection in split scenario is one half the RTT if there had been a single connection between the first sender and the last receiver (namely A and C). However this assumption has absolutely no effect on the analysis.

Hence substituting (3.20) in (3.22) we have

$$E[A] = RTT\left(\frac{(2+b)}{6} + \sqrt{\left(\frac{b+2}{6}\right)^2 + \frac{2b}{3}\left(\frac{(1-p)^2}{\left(1-(1-p)^2\right)}\right)} + 1\right) \tag{3.23}$$

3.2.4 Expressions for $E[Y]$ and $E[A]$

From (3.19) and (3.23) we have expressions for $E[Y]$ and $E[A]$,

$$E[Y] = \frac{(1-p)^2}{\left(1-(1-p)^2\right)} + \frac{(2+b)}{3b} + \sqrt{\left(\frac{b+2}{3b}\right)^2 + \frac{8}{3b}\left(\frac{(1-p)^2}{\left(1-(1-p)^2\right)}\right)}$$

and

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

Chapter 4

Timeout and Retransmission

4.1 Introduction

In this chapter we will study the behavior of TCP on the timing out of its retransmission timer. A TCP sender times out when packets are lost and less than three duplicate acknowledgments have been received and the retransmission timer expires. When a time out occurs, the sender waits for a certain duration of time, denoted by T_o, and then retransmits the lost packet again, if another time out occurs before this packet is successfully transmitted then the sender waits for a time $2T_o$ and then retransmits the lost packet. On successive time outs for the same packets, the time out duration is increased as $2T_o$, $4T_o$, $8T_o$, $16T_o$, $32T_o$ and then $64T_o$, after which it remains constant at $64T_o$. We will develop expressions for Q, $E[R]$ and $E\left[Z^{TO}\right]$.

4.2 Derivation of Q, $E[R]$ and $E\left[Z^{TO}\right]$

4.2.1 Derivation of Q

We will first derive an expression for Q.

Q is the probability that a loss indication is a TO. We define by $Q\prime(w)$, the probability that a loss indication in a window size w is a time out. If $A(w,k)$ denotes the probability that the first k packets are ACKed in a round of w packets sent in the penultimate round, then

$$A\left(w,k\right)=\left(1-p\right)^{2k}\left(1-\left(1-p\right)^{2}\right) \tag{4.1}$$

Also $C(n,m)$ is the probability that m packets are ACKed in sequence in the last round, where n packets are sent, then

$$C(n,m) = (1-p)^{2m}\left(1-(1-p)^2\right) \tag{4.2}$$

If the window size in the penultimate round is less than or equal to three, three duplicate acknowledgments can never be received. Hence the loss indication will always be a time out. But if the window size in the penultimate round is greater than 3, then a time out will occur in either of the following two cases:

Case1: Only two or less packets sent in the penultimate round are acknowledged, hence only another two packets may be sent in the last round and three duplicate ACKs can never be received.

Case2: Three or more packets sent in the penultimate round are acknowledged, but only two or less packets in the last round are successfully sent, then too three duplicate acknowledgments can never be received.

So

$$Q\prime(w) = \begin{cases} 1 & w \leq 3 \\ \sum_{k=0}^{2} A(w,k) + \left(\sum_{k=3}^{w-1} A(w,k) * \sum_{m=0}^{2} C(k,m)\right) & otherwise \end{cases} \tag{4.3}$$

or

$$Q\prime = \min(1, X) \tag{4.4}$$

where

$$X = \sum_{k=0}^{2} A(w,k) + \left(\sum_{k=3}^{w-1} A(w,k) * \sum_{m=0}^{2} C(k,m)\right) \tag{4.5}$$

We have

$$X = (1-(1-p))^6 + \left(1-(1-p)^6\right) * \left[(1-p)^6\left(1-(1-p)^{(2w-6)}\right)\right]$$

or

$$X = \left(1-(1-p)^6\right) * \left[1+(1-p)^6\left(1-(1-p)^{(2w-6)}\right)\right] \tag{4.6}$$

or

$$Q\prime(w) = \min\left(1,\ \left(1-(1-p)^6\right) * \left[1+(1-p)^6\left(1-(1-p)^{(2w-6)}\right)\right]\right) \quad (4.7)$$

Q, the probability that a loss indication is a time out, is given by

$$Q = \sum_{w=1}^{\infty} Q\prime(w) P\left[W = w\right]$$

or we write

$$Q = E\left[Q\prime\right]$$

or approximating it we have

$$Q = Q\prime\left(E[W]\right) \quad (4.8)$$

4.2.2 Derivation of $E[R]$

We next derive $E[R]$. We first determine how many packets are transmitted between two time outs in sequence. Once a time out occurs, the congestion window is reduced to one and one packet is retransmitted. Before the timer expires two other duplicate ACKs may be received as otherwise three duplicate acknowledgments would have been received, hence the window could be increased to a maximum of 3, before another time out occurs. So we arrive at a maximum number of $(1+2+3) = 6$. The number of packets sent between two time outs in sequence, for the same lost packet(s), would be in the closed interval $[1, 6]$. Let the average number of packets transmitted between two time outs in sequence (for the same lost packet(s)) be denoted by a constant d. So a sequence of k time outs occur, when there are $(k-1)d$ consecutive losses, followed by a successfully transmitted packet.

So

$$P[R = k] = \left(1-(1-p)^2\right)^{(k-1)d} * (1-p)^2 \quad (4.9)$$

$$E[R] = \sum_{k=1}^{\infty} kP[R = k] \quad (4.10)$$

or

$$E[R] = \sum_{k=1}^{\infty} k * \left(1-(1-p)^2\right)^{(k-1)d} * (1-p)^2$$

or

$$E[R] = (1-p)\sum_{k=1}^{\infty} k\left(1-(1-p)^2\right)^{(k-1)d}$$

or

$$E[R] = (1-p)^2\left[1+2\left(1-(1-p)^2\right)^{d}+3\left(1-(1-p)^2\right)^{2d}+...\right]$$

or

$$E[R] = (1-p)^2\left[\frac{1}{\left(1-\left(1-(1-p)^2\right)^{d}\right)}+\frac{\left(1-(1-p)^2\right)^{d}}{\left(1-\left(1-(1-p)^2\right)^{d}\right)^{2}}\right]$$

or

$$E[R] = (1-p)^2\left[\frac{\left(1-\left(1-(1-p)^2\right)^{d}\right)+\left(1-(1-p)^2\right)^{d}}{\left(1-\left(1-(1-p)^2\right)^{d}\right)^{2}}\right]$$

or

$$E[R] = \frac{(1-p)^2}{\left(1-\left(1-(1-p)^2\right)^{d}\right)^{2}} \tag{4.11}$$

For $d = 1$, (4.11) reduces to

$$E[R] = \frac{1}{(1-p)^2} \tag{4.12}$$

4.2.3 Derivation of $E\left[Z^{TO}\right]$

We next derive $E\left[Z^{TO}\right]$. We know that in any sequence of time outs, the first six have lengths T_o , $2T_o$, $4T_o$, $8T_o$, $16T_o$, $32T_o$ i.e. for $i = 1, 2, 3, 4, 5, 6$ the length of the time out is $2^{i-1}T_o$ and all the time outs immediately following it, have length of $64T_o$. A sequence of k timeouts will have a duration

$$L_k = \begin{cases} \left(2^k - 1\right) T_o & k \leq 6 \\ \left(63 + 64\left(k - 6\right)\right) T_o & k \geq 7 \end{cases} \tag{4.13}$$

We have

$$E\left[Z^{TO}\right] = \sum_{k=1}^{\infty} L_k P[R = k] \tag{4.14}$$

or

$$E\left[Z^{TO}\right] = \sum_{k=1}^{6} L_k P[R = k] + \sum_{k=7}^{\infty} L_k P[R = k] \tag{4.15}$$

We have

$$\sum_{k=1}^{6} L_k P[R = k] = \sum_{k=1}^{6} \left(2^k - 1\right) T_o * \left(1 - (1-p)^2\right)^{(k-1)d} * (1-p)^2$$

or

$$\sum_{k=1}^{6} L_k P[R = k] = (1-p)^2 T_o * \sum_{k=1}^{6} \left(2^k - 1\right) * \left(1 - (1-p)^2\right)^{(k-1)d} \tag{4.16}$$

We have

$$\sum_{k=7}^{\infty} L_k P[R = k] = \sum_{k=7}^{\infty} \left[63 + 64\left(k - 6\right)\right] T_o * \left(1 - (1-p)^2\right)^{(k-1)d} (1-p)^2 \tag{4.17}$$

or

$$\sum_{k=7}^{\infty} L_k P[R = k] = (1-p)^2 T_o \sum_{k=7}^{\infty} \left[63 + 64\left(k - 6\right)\right] * \left(1 - (1-p)^2\right)^{(k-1)d}$$

or

$$\sum_{k=7}^{\infty} L_k P[R = k] = (1-p)^2 \left(1 - (1-p)^2\right)^{6d} T_o \left[\frac{127 - 63\left(1 - (1-p)^2\right)^d}{\left(1 - \left(1 - (1-p)^2\right)^d\right)^2}\right] \tag{4.18}$$

Substituting (4.16) and (4.18) in (4.15) we have

$$E\left[Z^{TO}\right] = \sum_{k=1}^{6} L_k P[R=k] + \sum_{k=7}^{\infty} L_k P[R=k]$$

where

$$\sum_{k=1}^{6} L_k P[R=k] = (1-p)^2 T_o * \sum_{k=1}^{6} \left(2^k - 1\right) * \left(1-(1-p)^2\right)^{(k-1)d}$$

and

$$\sum_{k=7}^{\infty} L_k P[R=k] = (1-p)^2 \left(1-(1-p)^2\right)^{6d} T_o \left[\frac{127 - 63\left(1-(1-p)^2\right)^d}{\left(1-\left(1-(1-p)^2\right)^d\right)^2}\right] \tag{4.19}$$

Now if $d = 1$ (4.19) reduces to

$$E\left[Z^{TO}\right] = \sum_{k=1}^{6} L_k P[R=k] + \sum_{k=7}^{\infty} L_k P[R=k]$$

where

$$\sum_{k=1}^{6} L_k P[R=k] = (1-p)^2 T_o * \sum_{k=1}^{6} \left(2^k - 1\right) * \left(1-(1-p)^2\right)^{(k-1)}$$

and

$$\sum_{k=7}^{\infty} L_k P[R=k] = (1-p)^2 \left(1-(1-p)^2\right)^{6} T_o \left[\frac{64 + 63\,(1-p)^2}{(1-p)^4}\right] \tag{4.20}$$

4.2.4 Expressions for Q, $E[R]$ and $E\left[Z^{TO}\right]$

From (4.7), (4.8), (4.11) and (4.19) we have expressions for Q, $E[R]$ and$E\left[Z^{TO}\right]$

$$Q = Q\prime\left(E[W]\right)$$

where

$$Q\prime = \min\left(1,\ \left(1-(1-p)^6\right) * \left[1+(1-p)^6\left(1-(1-p)^{(2w-6)}\right)\right]\right)$$

$$E[R] = \frac{(1-p)^2}{\left(1-\left(1-(1-p)^2\right)^d\right)}$$

$$E\left[Z^{TO}\right] = \sum_{k=1}^{6} L_k P[R=k] + \sum_{k=7}^{\infty} L_k P[R=k]$$

where

$$\sum_{k=1}^{6} L_k P[R=k] = (1-p)^2 T_o * \sum_{k=1}^{6}\left(2^k-1\right) * \left(1-(1-p)^2\right)^{(k-1)d}$$

and

$$\sum_{k=7}^{\infty} L_k P[R=k] = (1-p)^2\left(1-(1-p)^2\right)^{6d} T_o \left[\frac{127-63\left(1-(1-p)^2\right)^d}{\left(1-\left(1-(1-p)^2\right)^d\right)^2}\right]$$

Chapter 5

Slow Start Behavior of TCP

5.1 Introduction

The slow start algorithm operates by observing that the rate at which new packets should be injected into the network is the rate at which the acknowledgments are returned by the receiver. When a new connection is established, cwnd is set to one and each time an ACK acknowledging new data is received, cwnd is increased by one. The sender can transmit up to the minimum of the cwnd and the receiver's advertised window. At some point the capacity of the connection would be reached and packets would start getting lost. This packet loss indication might be the receipt of a triple duplicate acknowledgement or the expiry of the retransmission timer. We will develop expressions for $E[L]$ and $E\left[Z^{SS}\right]$, the expected values for the number of packets sent during slow start and the duration of time the connection spends in slow start respectively.

5.2 Congestion Control Window Evolution

Let W denote TCP's congestion control window size. Since the connection is in slow start, W will be incremented by one each time an ACK acknowledging new data is received. TCP's slow start behavior is modeled in terms of rounds. A round starts with the back to back transmission of W packets, where W is the current transmission control window size. Once all the packets falling within the congestion control window have been sent, no other packet is sent until the first ACK is received for one of these W packets. The receipt of this ACK marks the

end of the current round and the beginning of the next round. The duration of the round is the same as the RTT and is assumed to be independent of the window size. Further, it is assumed that the time to send W packets is smaller than the round trip time. At the beginning of the next round, $W\prime$ new packets will be sent, where $W\prime$ would be the new size of the congestion control window, b is the number of packets that are acknowledged by a received ACK.

5.2.1 Derivation of $E[L]$

We denote by α_i the first packet lost in the slow start stage, i.e. till such time and including it, α_i packets have bene transmitted, X_i denotes the round where this loss occurs.

After packet α_i , a maximum of $W_i - 2$ more packets might be sent in an additional round before the current slow start stage comes to an end. This would be the case when the last packet transmitted in the penultimate round is the first packet that is lost. For each acknowledgement received acknowledging a packet sent in the penultimate round the window size is increased appropriately and packets are transmitted in the final round, if permitted. Thus a maximum of $L_i = \alpha_i + W_i - 2$ packets are sent in $X_i + 1$ rounds. We note that the slow start might also come to an end when the cwnd variable exceeds the ssthresh variable. If the ssthresh variable is set to an arbitrarily high value, then the chances of a loss occurring before the slow start stage comes to an end are greatly increased. This is also expected due to reasonable loss probability values.

It follows that

$$E[L] = E[\alpha] + E[W] - 2 \tag{5.1}$$

Next to derive $E[\alpha]$, we consider the random process $\{\alpha_i\}_i$, which is a sequence of independent and identically distributed random variables, since packets are lost independently of any other packets lost in any other rounds. The probability that $\alpha_i = k$ is equal to the probability that exactly $(k-1)$ packets are successfully acknowledged before a loss occurs, hence as derived above we have from (3.5)

$$E[\alpha] = \frac{1}{\left(1 - (1-p)^2\right)} \tag{5.2}$$

Substituting (5.2) in (5.1) we have

$$E[L] = \frac{1}{\left(1 - (1-p)^2\right)} + E[W] - 2 \tag{5.3}$$

We note that in the slow start sequence, the window size increases from 1 to W_i .

In other words

at the beginning of the first round the window size is 1

at the beginning of the second round it is 2

at the beginning of the third round the window size is 4, and so on.

The window size increases exponentially. So at the beginning of the last round the window size would be W_i

where

$$W_i = 2^{\left(\frac{X_i}{b}-1\right)},\ i = 1, 2, 3 \ldots \tag{5.4}$$

Note the assumption that $\frac{X_i}{b}$ is an integer

Since in slow start , L_i packets are transmitted, So the total number of packets sent would be

$$L_i = b\,(1) + b\,(2) + b\,(4) + ... + b\left(2^{\left(\frac{X_i}{b}-1\right)}\right) + \beta_i$$

or

$$L_i = b * \left[\sum_{k=0}^{\left(\frac{X_i}{b}-1\right)} 2^k\right] + \beta_i$$

where β_i is the number of packets sent in the last round

or

$$L_i = b\left[1 + 2 + 4 + ... + 2^{\left(\frac{X_i}{b}-1\right)}\right] + \beta_i$$

or

$$L_i = b * \left[2\left(\frac{X_i}{b}\right) - 1\right] + \beta_i \tag{5.5}$$

Using (5.4) we have

$$W_i = \frac{2^{\left(\frac{X_i}{b}\right)}}{2}$$

or

$$2W_i = 2^{\left(\frac{X_i}{b}\right)} \tag{5.6}$$

Substituting (5.6) in (5.5) we have

$$L_i = b * (2W_i - 1) + \beta_i$$

The expected value of L is given by

$$E[L] = b * (2E[W] - 1) + E[\beta] \tag{5.7}$$

Substituting (5.3) in (5.7) we have

$$\frac{1}{\left(1-(1-p)^2\right)} + E[W] - 2 = b * (2E[W] - 1) + E[\beta] \tag{5.8}$$

The number of packets sent in the last round have an uniform distribution between 0 and $2^{\left(\frac{X_i}{2}\right)} - 2$, or between 0 and $2W_i - 2$, hence

$$E[\beta] = E[W] - 1 \tag{5.9}$$

hence substituting (5.9) in (5.8) we have

$$\frac{1}{\left(1-(1-p)^2\right)} + E[W] - 2 = b * (2E[W] - 1) + E[W] - 1$$

or

$$2bE[W] - b = \frac{1}{\left(1-(1-p)^2\right)} - 1$$

or

$$2bE[W] = \frac{1 - \left(1-(1-p)^2\right)}{\left(1-(1-p)^2\right)} + b$$

or

$$2bE[W] = \frac{(1-p)^2}{\left(1-(1-p)^2\right)} + b$$

or

$$E[W] = \frac{1}{2b}\left[\frac{(1-p)^2}{\left(1-(1-p)^2\right)} + b\right]$$

or

$$E[W] = \left[\frac{1}{2b} * \left(\frac{(1-p)^2}{\left(1-(1-p)^2\right)}\right) + \frac{1}{2}\right] \tag{5.10}$$

Substituting (5.10) in (5.3) we have

$$E[L] = \frac{1}{\left(1-(1-p)^2\right)} - 2 + \frac{1}{2} + \frac{1}{2b}\left[\frac{(1-p)^2}{\left(1-(1-p)^2\right)}\right]$$

or

$$E[L] = \frac{1}{\left(1-(1-p)^2\right)} + \frac{1}{2b}\left[\frac{(1-p)^2}{\left(1-(1-p)^2\right)}\right] - \frac{3}{2} \tag{5.11}$$

5.2.2 Derivation of $E\left[Z^{SS}\right]$

Next we will derive an expression for $E\left[Z^{SS}\right]$. Let r_j be the duration of the j^{th} round of the slow start stage under consideration. Hence the duration of slow start stage is Z_i^{SS}

$$Z_i^{SS} = \sum_{j=1}^{X_i+1} r_j \tag{5.12}$$

Since r_j are considered to be random variables, assumed to be independent of the size of the congestion window and hence independent of the round number j. It has been assumed that the time required to send all the packets in any round is less than the round trip time.

It follows that

$$E\left[Z^{SS}\right] = (E[X]+1)\,E[r] \tag{5.13}$$

$E[r]$ is denoted by RTT, the average value of the round trip time for the connections in series under consideration, which in our case is the second connection of the split TCP connection. We again note the point that, for the sake of simplicity, we assume that the RTT of the second connection in split scenario is one half the RTT if there had been a single connection between the first sender and the last receiver (namely A and C). However this assumption has absolutely no effect on the analysis.

We next derive an expression for $E[X]$, Now X_i is the round where a loss occurs for the first time in the slow start sequence under consideration. For the X_i^{th} round to start, it must have occurred that packets in $X_i - 1$ rounds must have been transmitted successfully. In other words, that $2^{(X_i-2)}$ packets must have been transmitted successfully.

So $P[X_i = k]$ = probability that $\left(2^{k-1} - 1\right)$ packets have been transmitted successfully, before the round, where a loss occurs begins. So the expected value of X_i is

$$E[X] = \sum_{k=1}^{\infty} k \left((1-p)^2\right)^{\left(2^{(k-1)}-1\right)} \tag{5.14}$$

or

$$E[X] = \left[1 + 2(1-p)^2 + 3\left((1-p)^2\right)^3 + 4\left((1-p)^2\right)^7 + \ldots\right] \tag{5.15}$$

5.2.3 Expressions for $E[L]$ and $E\left[Z^{SS}\right]$

From (5.11), (5.13) and (5.14) we have expressions for $E[L]$ and $E\left[Z^{SS}\right]$

$$E[L] = \frac{1}{\left(1-(1-p)^2\right)} + \frac{1}{2b}\left[\frac{(1-p)^2}{\left(1-(1-p)^2\right)}\right] - \frac{3}{2}$$

$$E\left[Z^{SS}\right] = (E[X]+1)\,E[r]$$

where

$$E[X] = \sum_{k=1}^{\infty} k \left((1-p)^2\right)^{\left(2^{(k-1)}-1\right)}$$

Chapter 6

Receiver's Advertised Window Limitation

6.1 Introduction

The analysis carried out so far assumes that the congestion control window can grow unconstrained and that its value governs the data transmission. Until now we have considered the flow control as imposed by the sender while that imposed by the receiver has been neglected. In this chapter we take into account the receiver's advertised window size. At the beginning of the TCP flow establishment, the receiver advertises a maximum congestion window size. The sender can transmit up to the minimum of the congestion window and the receiver's advertised window. The congestion control window is flow control imposed by the sender, while the receiver's advertised window is flow control imposed by the receiver.

6.2 Effects of Window Limitation

Let W_{max} denote the window size as advertised by the receiver. Hence in a period without loss the window size may grow up to W_{max} , but may not grow beyond this value. We make a simplifying assumption in our analysis. We denote by W_{u1} the unconstrained congestion control window size when the connection is in congestion avoidance and by W_{u2} the unconstrained congestion control window size while the connection is in slow start. From (3.18) and (5.10) respectively we have

$$E[W_{u1}] = \frac{(2+b)}{3b} + \sqrt{\left(\frac{b+2}{3b}\right)^2 + \frac{8}{3b}\left(\frac{(1-p)^2}{(1-(1-p)^2)}\right)}$$

and

$$E[W_{u2}] = \left[\frac{1}{2b} * \left(\frac{(1-p)^2}{\left(1-(1-p)^2\right)}\right) + \frac{1}{2}\right]$$

We assume that when $E[W_{u1}] < W_{max}$, the receiver's advertised window limitation has negligible effect on the TCP throughput and hence $E[W] \approx E[W_{u1}]$, where $E[W]$ is the expected value of the congestion control window size when the connection is in congestion avoidance . But if $E[W_{u1}] \geq W_{max}$, then $E[W] \approx W_{max}$

Similarly when $E[W_{u2}] < W_{max}$, then $E[W] \approx E[W_{u2}]$, where $E[W]$ is the expected value of the window size when the connection is in slow start. But if $E[W_{u2}] \geq W_{max}$, then $E[W] \approx W_{max}$.

We will first analyze the effect on throughput of the receiver's advertised window size when the connection is in congestion avoidance and then we will study its effect when the connection is in slow start.

6.2.1 Congestion Avoidance

We first investigate the effect on throughput of the receiver's advertised window size, while the connection is in congestion avoidance. Consider a sequence of triple duplicate periods. During any of the triple duplicate periods the window size grows linearly up to W_{max} for U_i rounds and then remains constant for V_i rounds, at the end of which the current triple duplicate period comes to an end. Since the loss indication is assumed to be a triple duplicate, the window size would drop to $\frac{W_{max}}{2}$ and this process would continue. Hence we have

$$W_{max} = \frac{W_{max}}{2} + \frac{U_i}{b} \tag{6.1}$$

From (6.1) it follows that

$$E[U] = \frac{b}{2} W_{max} \tag{6.2}$$

Counting the number of packets sent in the i^{th} TDP we have

$$Y_i = \frac{U_i}{b}\left(\frac{W_{max}}{2} + W_{max}\right) + V_i W_{max} \tag{6.3}$$

From (6.2) and (6.3) we have

$$E[Y] = \frac{E[U]}{2}\left(\frac{3}{2}W_{max}\right) + E[V]W_{max}$$

or

$$E[Y] = \frac{b}{4}W_{max}\left(\frac{3}{2}W_{max}\right) + E[V]W_{max}$$

or

$$E[Y] = \frac{3b}{8}W_{max}^2 + E[V]W_{max} \tag{6.4}$$

Substituting (3.6) in (6.4) we have

$$\frac{(1-p)^2}{\left(1-(1-p)^2\right)} + W_{max} = \frac{3b}{8}W_{max}^2 + E[V]W_{max}$$

or

$$E[V] = \frac{(1-p)^2}{\left(1-(1-p)^2\right)W_{max}} + 1 - \frac{3b}{8}W_{max} \tag{6.5}$$

Since $X_i = U_i + V_i$ we have

$$E[X] = E[U] + E[V] \tag{6.6}$$

Substituting (6.2) and (6.5) in (6.6) we have

$$E[X] = \frac{b}{2}W_{max} + \frac{(1-p)^2}{\left(1-(1-p)^2\right)W_{max}} + 1 - \frac{3b}{8}W_{max}$$

or

$$E[X] = \frac{b}{8}W_{max} + \frac{(1-p)^2}{\left(1-(1-p)^2\right)W_{max}} + 1 \tag{6.7}$$

Hence we have

if $E[W_{u1}] < W_{max}$ then $E[W] \approx E[W_{u1}]$ then

$$E[Y] = \frac{(1-p)^2}{\left(1-(1-p)^2\right)} + \frac{(2+b)}{3b} + \sqrt{\left(\frac{b+2}{3b}\right)^2 + \frac{8}{3b}\left(\frac{(1-p)^2}{\left(1-(1-p)^2\right)}\right)}$$

and

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

and

$$Q = Q\prime(E[W])$$

else if $E[W_{u1}] \geq W_{max}$ then $E[W] \approx W_{max}$ then from (3.6) we have

$$E[Y] = \frac{(1-p)^2}{\left(1-(1-p)^2\right)} + W_{max} \tag{6.8}$$

and from (3.22) and (6.7)

$$E[A] = RTT\left(\frac{b}{8}W_{max} + \frac{(1-p)^2}{\left(1-(1-p)^2\right)W_{max}} + 2\right) \tag{6.9}$$

and

$$Q = Q\prime(W_{max}) \tag{6.10}$$

6.2.2 Slow Start

We next investigate the effect on throughput of the receiver's advertised window size while the connection is in slow start. During slow start the window size grows exponentially up to W_{max} for U_i rounds and then remains constant for V_i rounds, at the end of which the current slow start sequence comes to an end. The decrease in window size would depend on the type of loss indication, whether by the receipt of triple duplicate acknowledgments or by the expiry of the retransmission timer. Counting the number of packets sent while this sequence of slow start lasts, we have

$$L_i = \left(2^{\frac{U_i}{b}} - 1\right) + W_{max}V_i \tag{6.11}$$

where $\left(2^{\frac{U_i}{b}} - 1\right)$ is the number of packets sent in U_i rounds and $W_{max}E[V]$ packets are sent in V_i rounds. Further, the sequence of slow start comes to an end at the end of $X_i = U_i + V_i$ rounds.

From equation (5.4) we have

$$\left(2^{\frac{U_i}{b}-1}\right) = W_{max} \tag{6.12}$$

or

$$\left(2^{\frac{U_i}{b}}\right) = 2W_{max} \tag{6.13}$$

Substituting equation (6.13) in (6.11) we get

$$L_i = (2W_{max} - 1) + W_{max}V_i$$

It follows that

$$E[L] = (2W_{max} - 1) + W_{max}E[V] \tag{6.14}$$

Substituting equation (5.12) in (6.14) we have

$$\frac{1}{\left(1-(1-p)^2\right)} + \frac{1}{2b}\left[\frac{(1-p)^2}{\left(1-(1-p)^2\right)}\right] - \frac{3}{2} = (2W_{max} - 1) + W_{max}E[V]$$

or

$$E[V] = \frac{1}{\left(1-(1-p)^2\right)W_{max}} + \frac{1}{2bW_{max}}\left[\frac{(1-p)^2}{\left(1-(1-p)^2\right)}\right] - \frac{1}{2W_{max}} - 2 \tag{6.15}$$

From (6.12) we have

$$\left(2^{\frac{U_i}{b}-1}\right) = W_{max}$$

or

$$\frac{U_i}{b} - 1 = log_2 W_{max}$$

or

$$U_i = b\,(1 + log_2 W_{max})$$

or

$$U_i = f(W_{max}) \tag{6.16}$$

where

$$f(W_{max}) = b\,(1 + log_2 W_{max})$$

From equation (6.16) it follows that

$$E[U] = E[f(W_{max})]$$

approximating this we may write

$$E[U] = f(W_{max})$$

or

$$E[U] = b\,(1 + log_2 W_{max}) \tag{6.17}$$

Since

$$E[X] = E[U] + E[V]$$

we have

$$E[X] = b\,(1 + log_2 W_{max}) + \frac{1}{\left(1-(1-p)^2\right) W_{max}} + \frac{1}{2bW_{max}} \left[\frac{(1-p)^2}{\left(1-(1-p)^2\right)}\right] - \frac{1}{2W_{max}} - 2 \tag{6.18}$$

Hence we have
if $E[W_{u2}] < W_{max}$ then $E[W] \approx E[W_{u2}]$ then

$$E[L] = \frac{1}{\left(1-(1-p)^2\right)} + \frac{1}{2b}\left[\frac{(1-p)^2}{\left(1-(1-p)^2\right)}\right] - \frac{3}{2}$$

and

$$E\left[Z^{SS}\right] = (E[X]+1)\,RTT$$

where

$$E[X] = \sum_{k=1}^{\infty} k \left((1-p)^2\right)^{\left(2^{(k-1)}-1\right)}$$

else if $E[W_{u1}] \geq W_{max}$ then $E[W] \approx W_{max}$ then from (5.7) and (5.9) we have

$$E[L] = 2bE[W] + E[W] - 1 - b$$

or

$$E[L] = (W_{max}(2b+1)) - (b+1) \tag{6.19}$$

and

$$E[Z^{SS}] = (E[X]+1)\,RTT$$

where

$$E[X] = b\,(1 + log_2 W_{max}) + \frac{1}{\left(1-(1-p)^2\right) W_{max}} + \frac{1}{2bW_{max}} \left[\frac{(1-p)^2}{\left(1-(1-p)^2\right)}\right] - \frac{1}{2W_{max}} - 2$$

6.3 Complete Characterization

The complete characterization of TCP throughput is given by the following equations

6.3.1 Case I

if $E[W_{u1}] < W_{max}$ and $E[W_{u2}] < W_{max}$ then from equations (2.11), (3.19), (3.23), (4.7), (4.8). (4.11), (4.19), (5.11), (5.13) and (5.14)

$$B = \frac{E[Y] + Q * (E[R] + E[L])}{E[A] + Q * (E[Z^{TO}] + E[Z^{SS}])} \tag{6.20}$$

where

$$E[Y] = \frac{(1-p)^2}{\left(1-(1-p)^2\right)} + \frac{(2+b)}{3b} + \sqrt{\left(\frac{b+2}{3b}\right)^2 + \frac{8}{3b}\left(\frac{(1-p)^2}{\left(1-(1-p)^2\right)}\right)}$$

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

$$Q = Q\prime(E[W])$$

where

$$Q\prime = \min\left(1,\ \left(1-(1-p)^6\right) * \left[1 + (1-p)^6\left(1-(1-p)^{(2w-6)}\right)\right]\right)$$

$$E[R] = \frac{(1-p)^2}{\left(1-\left(1-(1-p)^2\right)^d\right)}$$

$$E\left[Z^{TO}\right] = \sum_{k=1}^{6} L_k P[R=k] + \sum_{k=7}^{\infty} L_k P[R=k]$$

where

$$\sum_{k=1}^{6} L_k P[R=k] = (1-p)^2 T_o * \sum_{k=1}^{6} \left(2^k - 1\right) * \left(1-(1-p)^2\right)^{(k-1)d}$$

and

$$\sum_{k=7}^{\infty} L_k P[R=k] = (1-p)^2 \left(1-(1-p)^2\right)^{6d} T_o \left[\frac{127 - 63\left(1-(1-p)^2\right)^d}{\left(1-\left(1-(1-p)^2\right)^d\right)^2}\right]$$

$$E[L] = \frac{1}{\left(1-(1-p)^2\right)} + \frac{1}{2b}\left[\frac{(1-p)^2}{\left(1-(1-p)^2\right)}\right] - \frac{3}{2}$$

$$E\left[Z^{SS}\right] = (E[X]+1)\,RTT$$

where

$$E[X] = \sum_{k=1}^{\infty} k \left((1-p)^2\right)^{\left(2^{(k-1)}-1\right)}$$

6.3.2 Case II

if $E[W_{u1}] \geq W_{max}$ and $E[W_{u2}] \geq W_{max}$ then from equations (2.11), (4.7), (4.11), (4.19), (5.13), (6.8), (6.9), (6.10), (6.18) and (6.19)

$$B = \frac{E[Y] + Q * (E[R] + E[L])}{E[A] + Q * (E\left[Z^{TO}\right] + E\left[Z^{SS}\right])} \tag{6.21}$$

where

$$E[Y] = \frac{(1-p)^2}{\left(1-(1-p)^2\right)} + W_{max}$$

$$E[A] = RTT\left(\frac{b}{8}W_{max} + \frac{(1-p)^2}{\left(1-(1-p)^2\right)W_{max}} + 2\right)$$

$$Q = Q\prime\left(W_{max}\right)$$

where

$$Q\prime = \min\left(1,\ \left(1-(1-p)^6\right) * \left[1 + (1-p)^6\left(1-(1-p)^{(2w-6)}\right)\right]\right)$$

$$E[R] = \frac{(1-p)^2}{\left(1-\left(1-(1-p)^2\right)^d\right)}$$

$$E\left[Z^{TO}\right] = \sum_{k=1}^{6} L_k P[R=k] + \sum_{k=7}^{\infty} L_k P[R=k]$$

where

$$\sum_{k=1}^{6} L_k P[R=k] = (1-p)^2 T_o * \sum_{k=1}^{6}\left(2^k - 1\right) * \left(1-(1-p)^2\right)^{(k-1)d}$$

and

$$\sum_{k=7}^{\infty} L_k P[R=k] = (1-p)^2\left(1-(1-p)^2\right)^{6d} T_o\left[\frac{127 - 63\left(1-(1-p)^2\right)^d}{\left(1-\left(1-(1-p)^2\right)^d\right)^2}\right]$$

$$E[L] = (W_{max}(2b+1)) - (b+1)$$

$$E[Z^{SS}] = (E[X]+1)\,RTT$$

where

$$E[X] = b\,(1 + log_2 W_{max}) + \frac{1}{\left(1-(1-p)^2\right)W_{max}} + \frac{1}{2bW_{max}}\left[\frac{(1-p)^2}{\left(1-(1-p)^2\right)}\right] - \frac{1}{2W_{max}} - 2$$

6.3.3 Case III

if $E[W_{u1}] \geq W_{max}$ and $E[W_{u2}] < W_{max}$ then from equation (2.11), (4.7), (4.11), (4.19), (5.11), (5.13), (5.14), (6.8), (6.9) and (6.10)

$$B = \frac{E[Y] + Q * (E[R] + E[L])}{E[A] + Q * (E[Z^{TO}] + E[Z^{SS}])} \tag{6.22}$$

where

$$E[Y] = \frac{(1-p)^2}{\left(1-(1-p)^2\right)} + W_{max}$$

$$E[A] = RTT\left(\frac{b}{8}W_{max} + \frac{(1-p)^2}{\left(1-(1-p)^2\right)W_{max}} + 2\right)$$

$$Q = Q\prime\left(W_{max}\right)$$

where

$$Q\prime = \min\left(1,\ \left(1-(1-p)^6\right) * \left[1 + (1-p)^6\left(1-(1-p)^{(2w-6)}\right)\right]\right)$$

$$E[R] = \frac{(1-p)^2}{\left(1-\left(1-(1-p)^2\right)^d\right)}$$

$$E\left[Z^{TO}\right] = \sum_{k=1}^{6} L_k P[R=k] + \sum_{k=7}^{\infty} L_k P[R=k]$$

where

$$\sum_{k=1}^{6} L_k P[R=k] = (1-p)^2 T_o * \sum_{k=1}^{6}\left(2^k - 1\right) * \left(1-(1-p)^2\right)^{(k-1)d}$$

and

$$\sum_{k=7}^{\infty} L_k P[R=k] = (1-p)^2 \left(1-(1-p)^2\right)^{6d} T_o \left[\frac{127-63\left(1-(1-p)^2\right)^d}{\left(1-\left(1-(1-p)^2\right)^d\right)^2}\right]$$

$$E[L] = \frac{1}{\left(1-(1-p)^2\right)} + \frac{1}{2b}\left[\frac{(1-p)^2}{\left(1-(1-p)^2\right)}\right] - \frac{3}{2}$$

and

$$E\left[Z^{SS}\right] = (E[X]+1)\,RTT$$

where

$$E[X] = \sum_{k=1}^{\infty} k\left((1-p)^2\right)^{\left(2^{(k-1)}-1\right)}$$

6.3.4 Case IV

if $E[W_{u1}] < W_{max}$ and $E[W_{u2}] \geq W_{max}$ then from equations (2.11), (3.19), (3.23), (4.7), (4.8). (4.11), (4.19), (5.13), (6.19) and (6.18)

$$B = \frac{E[Y]+Q*(E[R]+E[L])}{E[A]+Q*(E\left[Z^{TO}\right]+E\left[Z^{SS}\right])} \tag{6.23}$$

where

$$E[Y] = \frac{(1-p)^2}{\left(1-(1-p)^2\right)} + \frac{(2+b)}{3b} + \sqrt{\left(\frac{b+2}{3b}\right)^2 + \frac{8}{3b}\left(\frac{(1-p)^2}{\left(1-(1-p)^2\right)}\right)}$$

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

$$Q = Q\prime\left(E[W]\right)$$

where

$$Q\prime = \min\left(1,\ \left(1-(1-p)^6\right) * \left[1+(1-p)^6\left(1-(1-p)^{(2w-6)}\right)\right]\right)$$

$$E[R] = \frac{(1-p)^2}{\left(1-\left(1-(1-p)^2\right)^d\right)}$$

$$E\left[Z^{TO}\right] = \sum_{k=1}^{6} L_k P[R=k] + \sum_{k=7}^{\infty} L_k P[R=k]$$

where

$$\sum_{k=1}^{6} L_k P[R=k] = (1-p)^2 T_o * \sum_{k=1}^{6}\left(2^k-1\right) * \left(1-(1-p)^2\right)^{(k-1)d}$$

and

$$\sum_{k=7}^{\infty} L_k P[R=k] = (1-p)^2\left(1-(1-p)^2\right)^{6d} T_o \left[\frac{127-63\left(1-(1-p)^2\right)^d}{\left(1-\left(1-(1-p)^2\right)^d\right)^2}\right]$$

$$E[L] = (W_{max}(2b+1)) - (b+1)$$

$$E[Z^{SS}] = (E[X]+1)\,RTT$$

where

$$E[X] = b\,(1+log_2 W_{max}) + \frac{1}{\left(1-(1-p)^2\right)W_{max}} + \frac{1}{2bW_{max}}\left[\frac{(1-p)^2}{\left(1-(1-p)^2\right)}\right] - \frac{1}{2W_{max}} - 2$$

Chapter 7

Implementation details

7.1 Role of Simulation

Analysis provides a model of the Internet over which one has complete control. It brings with it a better understanding of the various parameters of the Internet. But often there is the danger of oversimplified assumptions, which lead to the key issues of the Internet being ignored or misestimated. An analytically developed model can be verified by one or more of the following: measurement, experimentation and simulation. Measurements are needed for reality checks to test the implicit assumptions. Experiments are important for the consideration of implementation issues. They help make clear comparisons with common implementations. Simulations allow the exploration of complicated scenarios. The complexity of Internet traffic makes simulations a very important tool. Strategies for developing meaningful simulations have been put forth in [9, 10]. We chose to empirically validate our expressions using simplified simulation scenarios, which abstract away all but the important aspects of the analysis.

7.2 Network Simulator - Version 2

7.2.1 Introduction

The equations specified by (6.20), (6.21), (6.22) and (6.23) provide an analytic characterization of a split TCP connection as a function of the packet loss rate in any connection, the average round trip time of the connections in series and the receiver's advertised window size. The expressions developed were empirically

validated with measurement data obtained from TCP connections simulated using the Network Simulator, version 2 (ns) [12]. Ns is a discrete event simulator targeted at networking research. Ns is an object oriented simulator, written in C++, with an OTcL interpreter as a front-end. Ns provides substantial support for simulation of TCP networks. But it should be noted that ns is not a polished and finished product, instead is the result of an on-going effort of research and development.

7.2.2 TCP Simulation

Running a TCP simulation requires creating and configuring the agent, attaching an application-level data source (a traffic generator) and starting the agent and the traffic generator. There are two major types of TCP agents in ns: one-way agents and two-way agents. The one way agent represents a one way data sender. They must peer with a *TCP sink* object. The base TCP sink object is responsible for returning ACKs to a peer TCP source object. It generates one ACK per packet received. The two way agent is symmetric in the sense that it represents both, a sender and a receiver. The two way experimental sender currently only supports a Reno form of TCP. The generation of SYN packets (and their ACKs) can be of critical importance in trying to model real-world behavior, when using many very short data transfers. This version of TCP currently defaults to sending data on the third packet of an initial three-way handshake, a behavior somewhat different than common real-world TCP implementations. Thus, this version of TCP sends data at a time somewhat earlier than typical implementations. The TCP agent does not generate any application data on its own, instead the simulation user can connect any traffic generation module to the TCP agent to generate data. Two applications are commonly used for TCP: FTP and Telnet. FTP represents bulk data transfer of large size and Telnet uses its data size randomly. The TCP agent supports various configurable variables such as the packet size, receiver's window size, initial slow start threshold value, etc.

7.2.3 Ns Limitations

TCP Agents

A simulator model of a real-world system is necessarily a simplification of the real-world system itself, hence there exist some limitations of the simulation model. Ns supports several versions of an abstracted TCP sender. These objects attempt to capture the essence of the TCP congestion and error control behaviors, but are not intended to be faithful replicas of real-world TCP implementations. They do

not contain a dynamic window advertisement, segment and ACK number computations are in units of packets, and there is no SYN/FIN connection establishment/teardown. Further, no data is ever transferred, for example, no checksums or urgent data. The receiver's advertised window size once set remains constant and cannot be changed during a connection.

Round Trip Time Estimation

Five variables are used to estimate the RTT and set the retransmission timer: rtt_, srtt_, rttvar_, tcpTick_, and backoff_. TCP initializes rttvar_ to $\frac{3}{tcpTick_}$ and backoff_ to 1. When any future retransmission timer is set, its timeout is set to the current time plus $\min(bt(a + 4v + 1), 64)$ seconds where b is the current backoff_ value, t is the value of the tcpTick_, a is the value of srtt_ and v is the value of rttvar_. RTT samples arrive with new acknowledgements. The RTT sample is computed as the difference between the current time and a *time echo* field in the ACK packet. When the first sample is taken, its value is used as the initial value for srtt_. Half the first sample is used as the initial value for rttvar_. For subsequent samples, the values are updated as follows:

$$srtt = \frac{7}{8}srtt + \frac{1}{8}sample \tag{7.1}$$

$$rttvar = \frac{3}{4}rttvar + \frac{1}{4}\left[sample - srtt\right] \tag{7.2}$$

7.3 TCP Reno

Our model is based on the TCP Reno implementation. Different implementations differ in the manner in which they perform congestion control. TCP Reno induces packet losses to estimate the available bandwidth in the network. TCP Reno increases its window on the receipt of acknowledgments. When it experiences a loss, it reduces its window to the reciprocal of its current size. This is called *additive increase and multiplicative decrease*. It has been shown that this algorithm leads to a fair allocation of bandwidth [11].

In the ns implementation of TCP Reno, the congestion window is increased by one packet per each new ACK received during slow start (when $cwnd_ < ssthresh_$) and is increased by $\frac{1}{cwnd_}$ for each new ACK received during congestion avoidance (when $cwnd_ \geq ssthresh_$). TCP Reno assumes a packet has been lost (due to

congestion) when it observes three duplicate ACKs or when a retransmission timer expires. If it is the latter then TCP Reno reacts by setting $ssthresh_$ to half of the current window size (the minimum of $cwnd_$ and $window_$) or 2, whichever is larger. It then initializes $cwnd_$ back to the value of the initial window. This will typically cause the TCP to enter slow start. TCP Reno implementation in ns also includes fast recovery, where the current congestion window is 'inflated' by the number of duplicate ACKs the TCP sender has received before receiving a new ACK. A *new ACK* refers to any ACK with a value higher than the highest seen so far. The TCP Reno agent, during slow start, sets the congestion window to half the current window size and resets $ssthresh_$ to match this value. We note that the TCP Reno described above represents a one way data sender. It must peer with a *TCP sink* object responsible for returning ACKs to the TCP Reno sender.

7.4 Simulations

7.4.1 Introduction

We simulated a split TCP connection using ns. It has already been pointed out in chapter 2, that in a two connection scenario, it would suffice to count the number of packets sent in the second connection. The number of packets still in transit in the first connection, when the observed interval comes to an end, would be negligible. If a split TCP connection between a sender A and an eventual receiver C were to be set up and if B were to denote the intermediate point, then B-C would denote the second TCP connection. We simulated the second TCP connection in ns.

7.4.2 Simulation Configuration

Running a TCP simulation requires setting up a topology, creating and configuring the agent, attaching an application level data source (a traffic generator) and starting the agent and the traffic generator. The simulations were carried out for different durations, spread over a few seconds to a few hours.

Topology

The first step in setting up the simulation was creating a topology. This involved creating the nodes and connecting the nodes to form links. To set up the simulation of the second TCP connection, two nodes (B and C) were created and a

bi-directional link was established between these two nodes. The duplex-link is essentially created from two simplex links, one from node B to node C and the other from node C to node B. The bandwidth and the delay on the link were set to various values to study their effect on the throughput. It should be noted that the delay represents the time required for a packet to traverse a link. A first-in-first-out (FIFO) queue was associated with the link.

TCP Agent

A TCP Reno agent was created and configured. The configuration parameters included the packet size, the window size, the initial value of the slow start threshold and the granularity of the clock. The configuration parameters were varied in different runs of the simulation to study the effect of network parameters on the throughput of the connection. This TCP Reno agent was peered to a base TCP sink object responsible for returning acknowledgments to the TCP Reno source object. It was initially configured to generate one acknowledgement per packet received. This was varied to study the effect of b, the number of packets acknowledged by each acknowledgement, on the throughput of the connection.

Error Model

An error model was implemented and configured to introduce packet losses into the simulation. The error model simulated link-level losses by dumping the packet to a drop target. The errors were generated from a simple model. The unit of error was specified in terms of packets rather than in bytes. The random variable for generating errors was set to have a uniform distribution from 0 to 1.

The probability that a packet gets lost in the second link, the connection under consideration, depends on the loss probability in the first link. p represents the probability that a packet gets lost on a link. Using this value the probability that a packet would get lost in the second connection was calculated and the same was set as the packet loss probability in the error module.

Simulated Applications

Applications sit on top of transport agents in ns. FTP was the application associated with the TCP agent. It simulates bulk data transfer. They work by advancing the count of packets available to be sent by a TCP transport agent. The actual

transmission of available packets is still controlled by TCP's flow and congestion control algorithms.

Trace Data

The trace data containing record of each individual packet, as it arrives, departs or is dropped at a link or a queue, was written onto files. Programs were written to analyze these trace files and to plot graphs and generate data to empirically verify the output obtained from ns against those predicted by our expressions.

Chapter 8

Results and Discussion

8.1 Introduction

The results obtained from the simulations were analyzed from two standpoints

1. To compare the throughput predicted by our expressions against that generated by ns
2. To compare the throughput predicted by our expression for a split connection against the throughput generated by ns for a single connection between the endpoints

8.2 Analysis

8.2.1 Comparison Against Ns

Since the idea is to compare the throughput as predicted by our expressions against those generated by ns, we will study the effect of varying various network parameters on the agreement of these two values. We define the agreement between the two values as follows

$$Agreement = \frac{Throughput\ as\ predicted\ by\ our\ expression}{Throughput\ generated\ by\ ns} \tag{8.1}$$

So, an Agreement of one would mean that the throughput predicted by our expression is exactly the same as that generated by ns and an Agreement of 1.5 would

imply that the throughput predicted by our expression is one and half times that generated by ns

Duration Vs. Agreement

A series of simulations were carried out to determine the effect, if any, of the duration of the simulation on the Agreement. We set up a three node split TCP connection. For preset values of packet loss rate and the round trip time, the duration was plotted on the X Axis and the Agreement was plotted on the Y-axis. The packet loss rate and the round trip time were varied over wide ranges and the Duration Vs. Agreement plots were drawn for the same.

We found that there was practically no effect of the duration of simulation on the Agreement. Shown below is one of the graphs plotted for reasonable values of the packet loss rate and the round trip time. Again, it should be noted that the round trip time refers to the round trip time of the second connection in series.

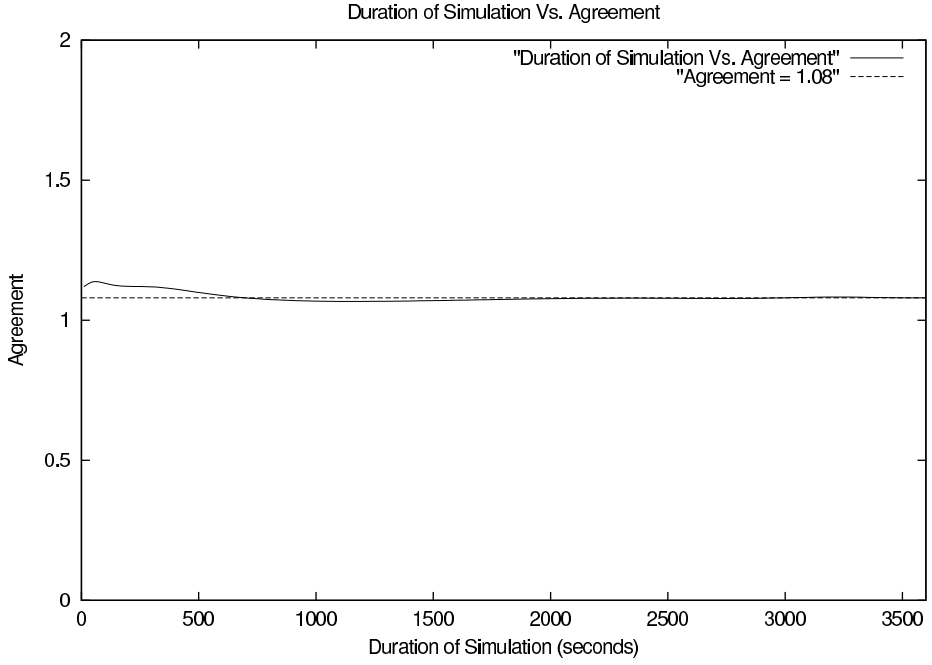

Figure 8.1: Plot of Duration of Simulation Vs. Agreement
packet loss probability = 0.005
RTT = 0.05 seconds

The Agreement over a wide range of simulation durations (10 seconds to one hour) was almost constant about 1.08. As above, the curve was almost horizontal for

different preset values of packet loss probability and round trip times, only that it was displaced vertically above or below depending on the exact values of the preset parameters. Thus, we arrive at the conclusion that the duration of simulation does not affect the Agreement.

We next studied the effect of varying various parameters, namely round trip time and packet loss probability, on Agreement. Instead of plotting a three dimensional graph (RTT, packet loss probability, Agreement) we decided to analyze the scenario with two two-dimensional graphs. One being a plot of the round trip time against Agreement for a varied range of preset packet loss probability values and the other a plot of the packet loss probability against Agreement for a wide range of preset round trip time values.

Round Trip Time Vs. Agreement

The plot of Round Trip Time Vs. Agreement was generated to study the effect of varying round trip time on the Agreement. This plot was drawn for a wide range of preset values for the packet loss probability. Shown below is one of the graphs plotted for a reasonable value of the packet loss probability.

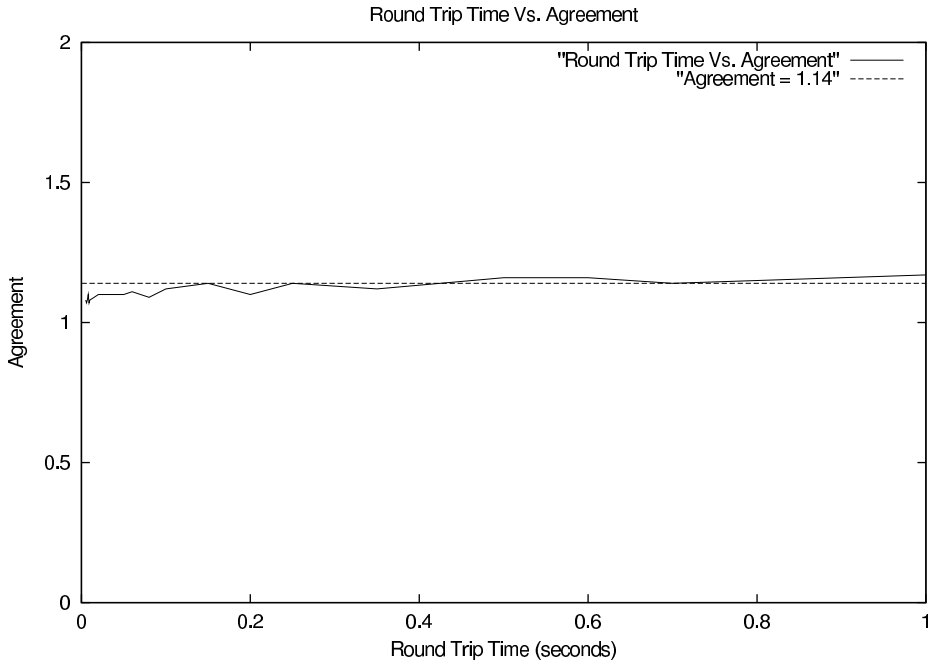

Figure 8.2: Plot of Round Trip Time Vs. Agreement
packet loss probability = 0.005

We note, that for the particular preset value of the packet loss probability the Agreement is almost constant around the $Agreement = 1.14$ line for a wide range of round trip times. Further, when the preset value of the packet loss probability was varied, it was found that the curve still remained almost horizontal, but it was displaced vertically between the Agreement range [0.9, 1.8] .Thus we reached the conclusion that varying the round trip time had negligible effect on Agreement between the throughput predicted by our equations and that generated by ns.

Packet Loss Probability Vs. Agreement

The plot of the Packet Loss Probability Vs. Agreement was drawn to analyze the effect that varying packet loss probability had on Agreement. Again as before plots were drawn for a wide range of preset round trip times. One of the graphs drawn for a reasonable preset round trip time is shown below.

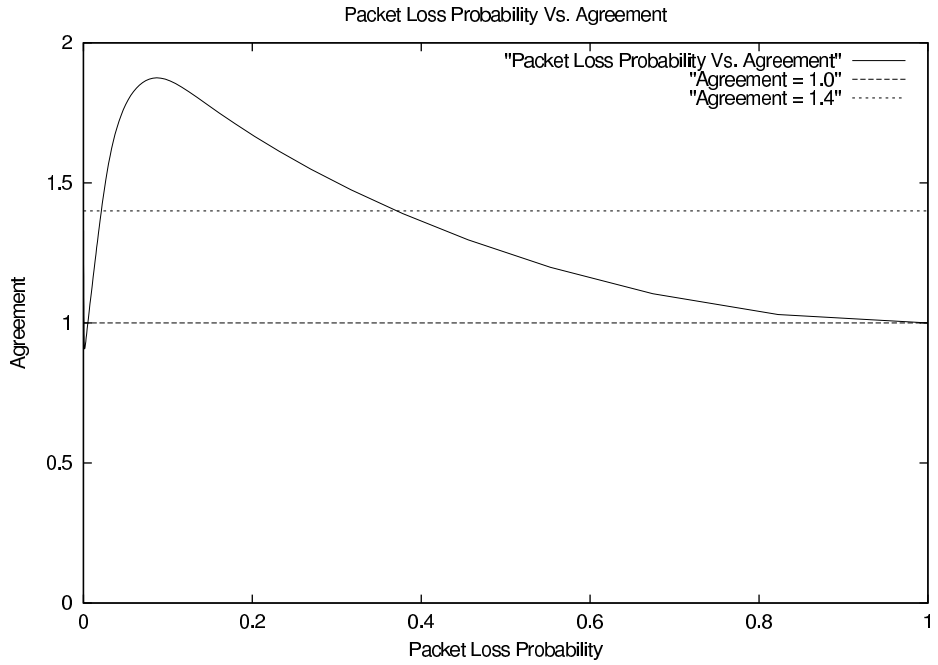

Figure 8.3: Plot of Packet Loss Probability Vs. Agreement
round trip time = 0.05 seconds

As observed previously, this plot was redrawn for a wide range of round trip time values. The shape of the graph was found to be almost the same for almost all the

plots drawn. We notice that for very small values of the packet loss probability ($p < 0.0002$), the Agreement lies in the range [1.16, 1.38]. As the probability increases to 0.0005, the Agreement moves towards 1. It bottoms out at 0.9 for a packet loss probability of 0.0018. From then on, as the loss probability increases to 0.1, the Agreement gradually moves from 0.9 to 1.9 and it levels off at 1.96 for a loss probability of 0.11. As the loss probability further increases all the way to 1, the Agreement slowly moves back to 1.

When the loss probability is really small ($p < 0.0002$), the number of packets transmitted is obviously very large. Since the numbers being dealt with are very large, a small shift might introduce a large error. Hence the Agreement is somewhat high. As the loss probability moves towards a more realistic (in terms of the Internet) value, the Agreement is almost 1. In our analysis, it has been found that expressions which do not model the slow start stage greatly over estimate the throughput. We have incorporated the slow start model in our equations. Hence the poor Agreement as the loss probability increases from 0.002 to 0.11 could be due to any or both of the following reasons

1. Our equations may not capture the entire essence of the evolution of the congestion control window, as the connection moves from slow start to congestion avoidance and then oscillates between congestion avoidance, fast retransmit and slow start
2. The values predicted by ns do not exactly simulate the TCP connection as various congestion control algorithms come into play

As the packet loss probability increases from 0.11 to almost 1, the Agreement starts getting better and and tends to 1.0 for extremely high loss rates. This might be due to the fact that as the loss probability increases there are more timeouts than triple duplicate loss indications. Further, the number of packets transmitted is significantly reduced. Hence even a large shift may not bring about a significant change in Agreement. Hence, even though the graph might give the impression that the agreement is very good, there might be significant descrepencies.

8.2.2 Comparison with a Single TCP Connection

In this section we compare the throughput predicted by our expression for a split connection against the throughput generated by ns for a single connection between the endpoints. As maintained throughout, a TCP connection between two endpoints is split into two at an intermediate point. For the sake of simplicity we have assumed the following:

1. The processing time of the packet at the intermediate point is negligible and that no packets are lost during the processing

2. The connection is split such that the round trip time of each of the two connections in series is one half the round trip time for the single connection. This was done for the sake of simplicity and clarity. It should be noted however that this has no bearings on the analysis carried out. But it is to be resolved for the sole purpose of comparing the performance of a split TCP connection with that of a single TCP connection.

We define $Increase$ to be an indication of the increase in throughput due to the splitting of the TCP connection.

$$Increase = \frac{throughput\ as\ predicted\ by\ our\ expression\ for\ a\ split\ connection}{throughput\ generated\ by\ ns\ for\ a\ single\ connection\ between\ endpoints} \tag{8.2}$$

Hence, if Increase were to be 2, then, that would mean that a split connection has a throughput twice that of a single connection between the endpoints. It has already been established that the round trip time does not affect the Agreement between theoretically predicted values and those generated by ns. Increase was plotted against a wide range of packet loss probabilities. However we generated these plots for a whole range of round trip times. The graph shown below shows the plot for a set of round trip times.

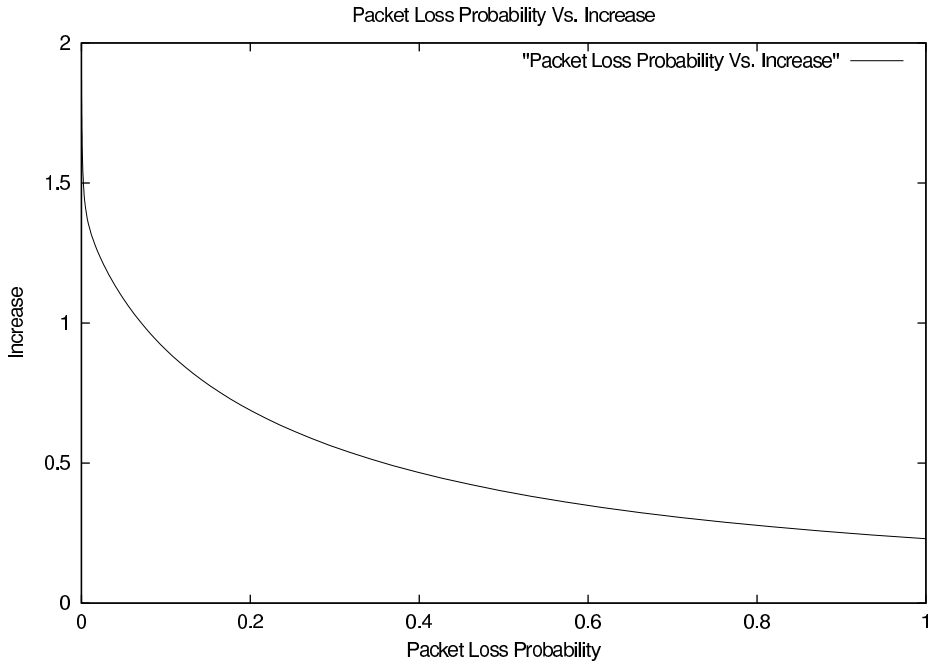

Figure 8.4: Plot of Packet Loss Probability Vs. Increase
round trip time for the full connection = 0.5 seconds
round trip time for the split connection = 0.25 seconds

We note from the graph that for reasonable loss probabilities of less than 0.05, the throughput predicted by the split connection is much greater than that of the single connection. However as the packet loss probability increases above 0.05 the throughput of the split connection drops significantly compared to the single connection case.

8.3 Conclusions

We have presented a model for a Split TCP connection implementing the Reno TCP. The model captures the essence of slow start, congestion avoidance and fast retransmit behavior of TCP. We developed an expression for the throughput of the split connection as a function of the packet loss rate in any connection, average round trip time of the connections in series and receiver's advertised window size.

Our model was empirically validated with ns. We noted in the ns simulations, that as the packet loss rate increased, the number of timeouts were significantly higher

than triple duplicate loss indications. There was some variation in the Agreement as the packet loss probabilities were increased beyond 0.11. It should be kept in mind that ns is the result of an on-going effort of research and development rather than a finished product. Since the number of timeouts significantly increased with the packet loss probability, the connection in such cases was found to spend considerable time in the slow start stage, thus making the slow start modeling an important part of the analysis. Our analysis includes the model for the slow start interval and hence provides a better model of the real world Internet.

In section 1.4 the point regarding rearranging the Internet was raised. It was noted that three important issues would need to be resolved before steps were taken in that direction. This work deals with one of them, the effect on throughput, performance in general, of the splitting of a TCP connection into one or more connections. Figure 8.4 illustrates the effect. For reasonable loss probabilities that exist in the Internet today, performance is significantly increased by the splitting of the connection. Since the answer to one of these questions is favorable it would be a worthwhile exercise to analyze at length the remaining two issues.

8.4 *N* Split Connections

Figure 2.1 illustrated a single TCP connection between two end points being split into $(n+1)$ connections in series. But for the sake of clarity and simplicity, instead of developing a model for $(n + 1)$ connections in series, we had considered the simplest case, consisting of two connections in series and had developed a model for the same. The equations specified by (6.20), (6.21), (6.22) and (6.23) provide an analytic characterization of a split TCP connection as a function of the loss rate in any connection, the average round trip time of the connections in series and the receiver's advertised window size. In this chapter we will comment on the case of a single TCP connection being split into $(n + 1)$ connections in series.

Though a generalized expression for n connections would be ideal, yet there is some wisdom in discussing the number of connections that a single TCP connection might be split into. As was noted in chapter 2, when there are two connections in series, then we only count the throughput of the second connection as the number of packets in transit in the first connection would be a negligible fraction of the throughput. But as the number of connections in series are increased, the cumulative count of the packets in transit in all but the last connection will increase to become a sizable factor, large enough not to be neglected in analysis. Further, the case of a single connection being split into two or three connections is not only a simpler case, but is in fact the more likely case. If the number of connections in

series were to increase then further analysis would have to be carried out to account for the packets in transit in the all but last intermediate TCP connections, when the observed interval comes to an end. These en-route packets have a certain chance of reaching their destination.

8.5 Future Work

A number of related areas can be identified for future work. Further analysis will need to be carried out for the general case of n TCP connections in series, wherein the packets in transit on the intermediate connections at the instant the observed interval comes to an end will no longer be neglected. In such cases, the assumption of a connection being split such that all the intermediate links have the same RTT would no longer hold and analyzing that in itself would be a significant research effort.

If the congestion window were to be modeled as a Markov chain a more precise expression for throughput could be developed. Once the Markovian arrival process has been developed to characterize losses on correlated paths, parameters would need to be identified. Further, our assumption that once a packet is lost in a round, then all the others packets lost in that round are lost too could be relaxed to incorporate a loss distribution function.

Three considerations were raised in section 1.4, which would need to be resolved satisfactorily before rearranging the Internet is carried out. We have addressed one of them, namely the effect on throughput of splitting a TCP connection. Further work will need to be done on working out the exact mechanics of carrying out this rearrangement and the overheads involved in the same.

Appendix A

Following is a table listing all the variables, their meaning and their values:

Number	Variable	Explanantion	Defining Equation Number
1	$E[M]$	Expected value of the number of packets sent in the TCP connection	(2.6)
2	$E[S]$	Expected value of the duration of the TCP connection	(2.7)
3	$E[n]$	Expected value of the number of TDPs in a sequence of TDPs	(2.10)
4	Q	Probability that the loss indication ending a TDP is a time-out	(2.10) and (4.8)
5	B	Throughput for a split TCP connection as a function of the packet loss rate, RTT and the receiver's advertised window size	(2.11), (6.20), (6.21), (6.22) and (6.23)
6	$E[Y]$	Expected value of the number of packets sent while the connection is in congestion avoidance	(3.19)
7	$E[\alpha]$	Expected value of the number of packets sent, until and including, the first lost packet while the connection is in congestion avoidnace and also while the connection is in slow start	(3.5) for congestion avoidance and (5.2) for slow start

8	$E[\beta]$	Expected value of the number of packets sent in the final round, while the connection is in congestion avoidance and also while it is in slow start	(3.14) for congestion avoidance and (5.9) for slow start
9	$E[W]$	Expected value of the congestion control window size at the end of a sequence of TDPs and also the congestion control window size at the end of a slow start sequence	(3.18) for congestion avoidance and (5.10) for slow start
10	$E[X]$	Expected value of the number of rounds in a TDP and also the number of rounds in a slow start sequence	(3.20) for congestion avoidance and (5.14) for slow start
11	$E[r]$	The round trip time of the second connection	(3.22)
12	$E[A]$	Expected value of the duration of the time the connection spends in congestion avoidance	(3.23)
13	$A(w,k)$	Probability that the first k packets are acknowledged in a round of w packets, sent in the penultimate round	(4.1)
14	$C(n,m)$	Probability that m packets are acknowledged in sequence in the last round, where n packets are sent	(4.2)
15	$Q\prime(w)$	Probability that a loss indication in a window size w is a timeout	(4.7)
16	$E[R]$	Expected value of the number of packets transmitted in a sequence for timeouts	(4.11)
17	$E\left[Z^{TO}\right]$	Expected value of the duration of a sequence of timeouts	(4.19)
18	$E[L]$	Expected value of the number of packets sent while the connection is in slow start	(5.11)
19	$E\left[Z^{SS}\right]$	Expected value of the duration of the time the connection spends in slow start	(5.13) and (5.14)

Bibliography

[1] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, 'Modeling TCP throughput: A Simple Model and it's Empirical Validation', ACM SIGCOMM, September 1998

[2] E. Altman, K. Avrachenkov and C. Barakat, 'A Stochastic Model of TCP/IP with Stationary Random Losses', SIGCOMM, 2000

[3] A. Bakre and B.R. Badrinath, 'I-TCP: Indirect TCP for Mobile Hosts, Proc. Second Usenix Symp. on Mobile and Location-Independent Computing, April 1995

[4] A. Bakre and B.R. Badrinath, 'Implementation and Performance Evaluation of Indirect TCP', IEEE Transactions on Computers, March 1997

[5] A.S. Tanenbaum, 'Computer Networks', Prentice Hall PTR, 1996

[6] W. R. Stevens, 'TCP/IP Illustrated, Volume 1, The Protocols', Addition Wesley, 1994

[7] I. F. Akyildiz, G. Morabito and S. Palazzo, 'Research Issues for Transport Protocols in Satellite IP Networks', IEEE Personal Communications, June 2001

[8] H. Balakrishnan, V. Padmanabhan, S. Seshan and R.H. Katz, 'A Comparison of Mechanisms for Improving TCP Performance over Wireless Links', IEEE/ACM Transactions on Networking, December 1997

[9] S. Floyd and V.Paxon, 'Difficulties in Simulating the Internet', IEEE/ACM Transactions on Networking, August 2001

[10] S. Floyd and V. Paxon, 'Why We Don't Know How to Simulate The Internet', Proc. of the 1997 Winter Simulation Conference, 1997

[11] J. Mo, R. J. La, V. Anantharam and J. Walrand, 'Analysis and Comparison of TCP Reno and Vegas', Proc. IEEE INFOCOM, 1999

[12] S. MCanne and S. Floyd, 'ns-LBL Network Simulator', 1997. Obtain via http://www.isi.edu/nsnam/ns/

[13] J. Madhavi and S. Floyd, 'TCP-Friendly Unicast Rate Based Flow control', Note sent to end2end-interest mailing list, January 1997

[14] P.L. Meyer, 'Introductory Probability and Statistical Applications', Addison-Wesley, 1970

[15] R. Nelson, 'Probability, Stochastic Processes, and Queuing Theory', Springer-Verlag, 1995

[16] L.B.W. Jolley, 'Summation os Series', Dover Publications, 1961

[17] V. Jacobson, 'Congestion Avoidance and Control', Proc. of SIGCOMM, November 1988

[18] J. Postel, 'Transmission Control Protocol', RFC793, September 1981

[19] M. Allman, V. Paxson and W. Stevens, 'TCP Congestion Control', RFC 2581, April 1999

[20] J.K. Ousterhout, 'Scripting:Higher-Level Programming for the 21st Century', IEEE Computer, March 1998

[21] A.C. Snoeren, H. Balakrishnan and M.F. Kaashoek, 'Reconsidering Internet Mobility', Proc. HotOS VIII, Elmau/Oberbayern, Germany, May 2001

[22] A.C. Snoeren and H. Balakrishnan, 'An End-to-End Approach to Host Mobility', Proc. 6th ACM MOBICOM, Boston, MA, August 2000

[23] A.C. Snoeren, H. Balakrishnan and D.G. Andersen, 'Fine-Grained Failover Using Connection Migration', Proc. 3rd Usenix Symp. on Internet Technologies and Systems, San Francisco, CA, March 2001

[24] D.G. Andersen, H. Balakrishnan, F. Kaashoek and R. Morris, 'The Case for Resilient Overlay Networks', Proc. SOSP 2001, Banff, Canada, October 2001

[25] P. Rodriguez, S. Sibal and O. Spatscheck, 'TPOT: Translucent Proxying of TCP', Technical Report TR 00.4.1, AT&T Research Labs, 2000

[26] D.R. Cheriton and M. Gritter, 'TRIAD: A Scalable Deployable NAT-Based Internet Architecture', March 2000

[27] F. Sultan, K. Srinivasan, L. Iftode, 'Transport Layer Support for Highly-Available Network Services', Rutgers University Technical Report, DCS-TR-429, January 2001

[28] S. Floyd and V. Jacobson, 'Random Early Detection Gateways for Congestion Avoidance', IEEE/ACM Transactions on Networking, August 1993

[29] K. Fall and S. Floyd, 'Simulation-based Comparisons of Tahoe, Reno, and SACK TCP', Computer Communications Review, July 1996

[30] F. Halsall, 'Data Communications, Computer Networks, and Open Systems', Addison-Wesley, 1992

[31] D. Comer, 'Internetworking with TCP/IP, Principles, Protocols and Architecture, Volume 1', Prentice Hall, 1995

[32] S.H. Low, F. Paganini and J.C. Doyle, 'Internet Congestion Control', To appear in IEEE Control Systems Magazine, February 2002

[33] D.E. Comer and J.C. Lin, 'Probing TCP Implementations', Proc. of USENIX Summer Conference, 1994

[34] R. Braden, 'Requirements for Internet Hosts – Communication Layers', RFC 1122, October 1989