MASTER THESIS

-

End-to-end TCP Performance in a Single- and Multi-domain Environment

Ran Yang
Student number: 1327933
Date: March 24, 2005
Business Mathematics and Informatics

Supervisors TNO ICT:
dr.ir. R.E. Kooij
prof.dr. J.L. van den Berg

Supervisors Free University Amsterdam:
prof.dr. R. van der Mei
dr. S. Bhulai

TNO

vrije Universiteit faculty of sciences amsterdam

Acknowledgements

This report has been written as the final assignment to achieve my Master of Sciences degree at the Department of Business Mathematics and Informatics at the Free University Amsterdam. This work is the result of a seven months internship at TNO ICT in Delft. Standing on the threshold of a new stage in life, I can look back at a fantastic and also very instructive period at TNO ICT.

These months of research and also the writing of this report could never have been successful without the help and support of several people. Therefore, I would like to express my gratitude to them for their support and constant advice.

During my stay at TNO ICT, I have received excellent support from several advisors. First of all I would like to express my deep appreciation and gratitude to my main research supervisor dr.ir. Rob Kooij for his constant encouragement and guidance during the course of this research and discussions on my research work. He gave me valuable suggestions on numerous occasions. Besides, I would like to thank prof.dr. Hans van den Berg for his supervision and comments on this project. Furthermore, I am grateful to ir. Pieter Venemans for helping me to solve the problems in the simulation program NS.

Besides the guidance from TNO ICT, I could always rely on the advice of prof.dr. Rob van der Mei, who supervised me on behalf of the VU. He provided me with useful input and moreover his infectious enthusiasm has certainly had its positive effect on my work. His critical reviews of my work have been very helpful in the realisation of this final report. Furthermore, I would like to thank my second supervisor dr. Sandjai Bhulai of the VU, whose useful comments have certainly contributed to an improved quality of this work.

I am also indebted to all office mates, without mentioning names, for the period we have spent together.

And I would like to express my sincere appreciation to my foster parents, Jan van Breemen and Edith Klinkhamer, for taking care of my son Chengji Riemer Zhao during my whole internship.

Finally my thanks go to my husband Dongsheng Zhao for his patience, understanding, and encouragement during the last research and thesis writing.

Ran Yang,

Delft, March 2005

Contents

1 Introduction

1.1 Background and motivation

Internet is playing an increasingly important role in daily life nowadays. In the past, the Internet was mainly used for sending an email or for finding documents. With the rise of the Internet and broadband communication technology, this is no longer the case. Internet is being designed to offer a wide range of services. People use Internet as a medium for on-line banking, for gaming, to order the latest books or CDs on-line, to download movies, etc. Because of the increasing possibilities of the Internet and the enormous growth of the number of Internet users, sufficient bandwidth is required. Unfortunately, while available bandwidth is increasing, the demand for that bandwidth is growing even faster. As a result, congestion appears in the data networks and leads to transmission delays, in many cases even to packet loss. The consequence is that the information which end users ask for experiences high delay or is incomplete.

To identify possibilities to prevent such kind of quality degradation, it is necessary to model and predict the performance of Internet applications. Internet consists of a variety of inhomogeneous network domains. Each network domain has its own characteristics, such as the packet loss probability (*p*) and the Round Trip Time (*RTT*). All information the clients request is assembled in packets and then transported over a multi-domain network that consists of a cascade of network domains. The greater part (>80%) of the data traffic over the Internet is regulated by the Transmission Control Protocol (TCP) (see e.g. [1], [2], [3]). The main performance measures to evaluate TCP are throughput or – from a user's point of view – simply the "speed of the Internet", and the transmission delay that includes the time it takes to make something appear on the computer screen after a link is clicked, and the download time of a web page or data file. Those two kinds of transmission delays are called in our thesis the response time (*RT*) and the total download time (*TDT*) respectively.

In the project EQUANET (End-to-End Quality of Service in Next-Generation Networks) [4], one of the key questions is how end-to-end QoS guarantees for a particular service can be derived from performance parameters of the involved network domains. In order to deal with this question, we establish in our work an analytical model for end-to-end TCP performance.

1.2 Research topic

Many studies have been investigating performance models for TCP. Recent work of Padhye et al. [5] develops an analytic characterization of the steady state throughput as a function of the packet loss rate (*p*) and the round trip time (*RTT*) for a TCP flow with an unlimited amount of data to send. According to [5], comparing to the TCP throughput models in [6], [7], and [8], this model is able to more accurately predict TCP throughput and is accurate over a wider range of loss rates. But the Padhye model does not take the behaviour of TCP's slow start and fast recovery mechanism into account. Therefore we attempt in our work to extend that model by considering the impacts of those two mechanisms. Comparing the throughput depicted by both models to the outcomes obtained from the simulation in ns-2 [9], we conclude that the extended model in general is more accurate.

Based on the goodput model of Padhye [10], Cardwell et al [11] propose a model for the mean total data download time (*TDT*). This model is quite accurate in predicting

the mean *TDT* under the assumption that losses happen only in the direction from the server to the client and losses in one round are independent of the losses in any other round. According to [5], [10], and [11], a *round* is the period of time between the departure of the first segment of the current window and the arrival of its acknowledgement (*ACK*). The value of the *RTT* is close to the duration of a *round.* But in real life, the ACKs, i.e., packets in the direction from the client to the server, may also be dropped. Therefore, we extend the Cardwell model by considering packet losses occurring in both directions. Apart from that, we investigate how to apply the Cardwell model to predict the mean TDT in the case that packet losses are correlated and the loss pattern is simulated in the two-state Gilbert loss model.
Cardwell's TDT model contains the model for predicting the TCP connection establishment time wherein it assumes that, when the SYN/ACK packet is lost, the receiver retransmits this lost packet. But we think it is more realistic that the sender retransmits the SYN packet when it does not receive the SYN/ACK packet after the retransmission time-out (*RTO*). In addition, the Cardwell TCP connection establishment model just focuses on the time taken to send *two* packets successfully, one by the sender and the other by the receiver. But the *RT* is the duration of sending four packets successfully. Therefore we cannot use the Cardwell TCP connection establishment model for predicting the mean *RT.* Thus, in our work, we develop a new *RT* model.
The TDT model and the RT model mentioned above are simply modelled in a single-domain environment. In a single-domain environment, a packet can be delivered by the server to the client over just one network domain. But in fact, the Internet consists of a variety of heterogeneous network domains. In most situations, before a data packet is delivered, it is transferred over a cascade of network domains. Therefore in our work, those two models are further extended in a multi-domain environment.

1.3 Overview of this report

In Chapter 2 we provide a description of TCP and discuss its performance relevant mechanisms. Chapter 3 discusses the Padhye model with our extension of that model. In Chapter 4 an analytical model for the response time in a single- and multi-domain environment is developed. Chapter 5 extends the model of Cardwell. In Chapter 6, the TDT model in a multi-domain environment is described. Subsequently, in Chapter 7 we discuss the feasibility of the Cardwell TDT model when the packet losses are correlated. All models developed in Chapter 3 to Chapter 7 are validated with ns-2 simulations, and are found to match very well. Finally, we present our conclusions and recommendations in Chapter 8.

2 Transmission Control Protocol

In this chapter we present in Section 2.1 the general characteristics of the TCP/IP suite and then in Section 2.2 we present details of the basic mechanisms in the TCP/IP suite which involve the aspects of data transfer. In Section 2.3, the Retransmission Time Out (*RTO*) is described and finally in Section 2.4 we briefly discuss several TCP versions that have become standardized.

2.1 The TCP/IP suite

The TCP/IP suite is a set of network protocols used widely in Internet nowadays. It consists of four layers. We refer to [12] for more details. Besides the well-known TCP and IP protocols, there are still more protocols apart form the four-layer structure. This layered representation brings forward the term protocol stack which is synonymous with protocol suite. With the help of clear interfaces between each layer, interconnection of various kinds of networks becomes possible. The four-layer structure is illustrated in Figure 2-1.

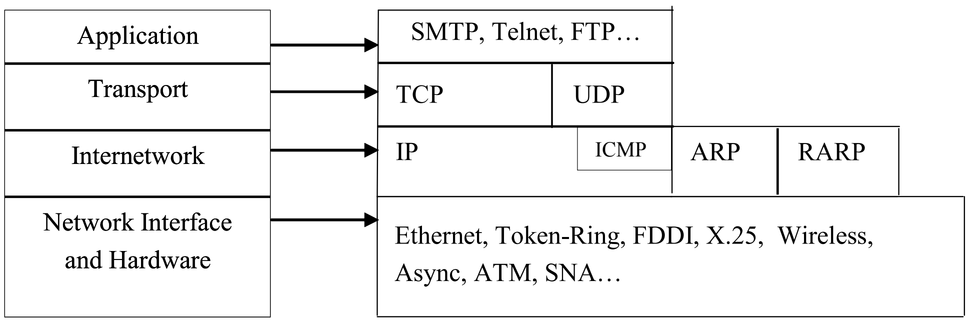

Application
Transport
Internetwork
Network Interface and Hardware
SMTP, Telnet, FTP…
TCP
UDP
IP
ICMP
ARP
RARP
Ethernet, Token-Ring, FDDI, X.25, Wireless, Async, ATM, SNA…

Figure 2-1: Four-layer structure of the TCP/IP suite.

At the base of the TCP/IP protocol stack is the Network Access layer. As described in [13], this layer manages all the services and functions necessary to prepare the data for the physical network. There is a lot of variation in the physical network, such as Ethernet, Token Ring, etc. All different types have their own conventions. Therefore the Network Access layer defines the procedures for interfacing with the network hardware and accessing the transmission medium.
The layer above the Network Access layer is the Internetwork layer. The best known TCP/IP protocol at the Internetwork layer is the Internet Protocol (*IP*), which provides the basic packet delivery service for all TCP/IP networks. The IP protocol implements a system of logical host addresses called IP addresses. The IP addresses are used by the Internetwork and higher layers to identify devices and to perform Internetwork routing. The Address Resolution Protocol (*ARP*) enables IP to identify the physical address that matches a given IP address.
The protocol layer just above the Internetwork layer is the Transport layer. It is responsible for the end-to-end data integrity. The two most important protocols employed at this layer are the Transmission Control Protocol (*TCP*) and the User Datagram Protocol (*UDP*). TCP provides full-duplex connections and reliable service. Before two application processes can start sending data to each other, they must establish a TCP connection between them. To guarantee the delivery of data, TCP uses

the acknowledgement mechanism to check if the transmitted data has been received correctly by the receiver. Acknowledgements (*ACK*s) are generated only by the receiver and only after the data packets are received correctly. When the transmission results in an error, TCP ensures that data is retransmitted while at the same time the data transmission rate is regulated. The transmission error can be detected in two ways: after a given retransmission timeout (*RTO*): the expected ACK is not received by the transmitter before the *RTO* runs out; or the transmitter receives triple duplicated acknowledgments from the receiver asking for the same data packet.

The unit of data that TCP sends to IP is called a TCP segment. The maximum segment size (*MSS*) is the largest ‘chunk’ of data that TCP will send to the other end. A typical value for the *MSS* is 1024 bytes.

When error detection is not required, UDP is the best choice for regulating data transmission. The reason is that UDP enhances network throughput at the Transport layer by providing unreliable datagram service (connectionless).

At the top of the TCP/IP suite is the Application layer. The Application layer includes all processes that use the Transport layer protocols to deliver data. There are many kinds of applications protocols. Most provide user services, and new services are always being added to this layer. The most widely known and implemented applications protocols are Telnet, FTP, SMTP, and HTTP.

2.2 TCP mechanisms related to data transfer

2.2.1 Connection establishment

TCP is a connection-oriented protocol. Every successful TCP connection begins with a “three-way handshake” in which the endpoints exchange initial sequence numbers. Sequence numbers are used to coordinate which data has been transmitted and received. The “three way handshake” can be described as follows.

1 The initiating side (normally called the client) performs an active open. It sends a segment with the first SYN flag set and the proposed initial sequence number in the sequence number field (seq = X). This is segment 1.

2 The receiver (normally called the server) performs a passive open. When it receives segment 1, it returns a segment (segment 2) with both the SYN and ACK flags set with the sequence number field set to its own assigned value for the reverse direction (seq = Y) and acknowledge field of X + 1 (ack = X + 1).

3 On the receipt of segment 2, the initiating side knows that the connection has been successfully established. It confirms this by sending a segment (segment 3) with just the ACK flag set and an acknowledgement field of Y + 1.

These three segments complete the connection establishment. Figure 2-2 illustrates the procedure.

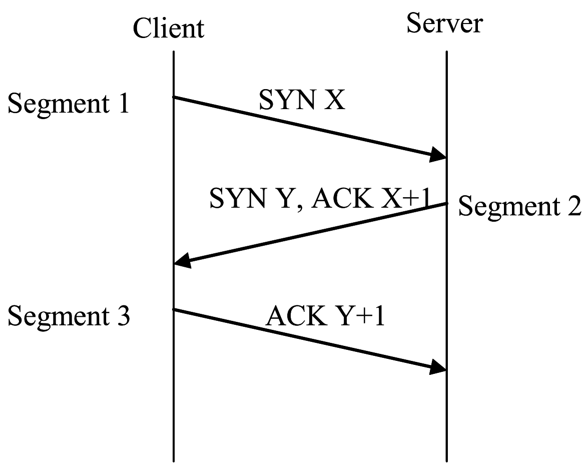


Figure 2-2: “Three-way handshake” in TCP connection establishment.

Per-connection state of the TCP, the sender side determines the congestion window (*cwnd*), that is the limitation of the amount of data the sender can transmit into the network before receiving an acknowledgment (ACK), and the receiver side determines the receiver’s advertised window (*rwnd*), that is the limitation of the amount of outstanding data.

2.2.2 *TCP Congestion Control*

After the TCP connection is established, *cwnd* and *rwnd* will be determined. The minimum of *cwnd* and *rwnd* governs data transmission. During the data transmission, congestion can occur when data arrives on a big pipe (a fast LAN) and gets send out on a smaller pipe (a slower WAN). Congestion can also occur when multiple input streams arrive at a router whose output capacity is less than the sum of the inputs. Modern implementations of TCP contain four intertwined congestion control algorithms: slow start, congestion avoidance, fast retransmit and fast recovery. These congestion control algorithms are devised by Van Jacobson ([14] and [15] provide the details).

2.2.2.1 *Slow Start (SS)*

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 not only at the beginning of a transmission, but also after repairing loss detected by RTO.

When the data transmission enters into the slow start mode, the *cwnd* is initialized to one segment. Each time an ACK is received, the *cwnd* is incremented from one to two, and two segments can be sent. When each of those two segments is acknowledged, the congestion window is increased to four. This provides an exponential growth if the receiver does not delay its ACKs. The sender can transmit up to the minimum of the congestion window and the advertised window.

Slow start ends when *cwnd* exceeds a certain threshold value (*ssthresh*) or when congestion is observed.

2.2.2.2 *Congestion Avoidance (CA)*

At some point when the process capacity limit of an intervening router is reached, packets can be dropped. CA is a way to deal with the lost packets. The assumption of the algorithm is that packet loss caused by damage is very small (much less than 1%), therefore the loss of a packet signals congestion somewhere in the network.
CA and SS are independent algorithms, but in practice they are implemented together. The combined algorithm operates as follows,

- According to [16], initialization for a given connection sets *cwnd* to one segment and *ssthresh* to 65535 bytes.

- When congestion occurs (indicated by a timeout or the reception of duplicate ACKs), one-half of the current window size (the minimum of *cwnd* and the receiver's advertised window, but at least two segments) is saved in *ssthresh*,
$$\mathrm{ssthresh} = \max\left(\frac{\min(\mathrm{cwnd}, \mathrm{rwnd})}{2}, 2\right).$$
  Additionally, if the congestion is indicated by a RTO, *cwnd* is set to one segment, i.e., it enters SS.

- When new data is acknowledged by the other end, we increase *cwnd*, but the way it increases depends on whether SS or CA is performed. If *cwnd* ≤ *ssthresh*, SS is performed; otherwise TCP is in CA.

CA dictates that *cwnd* be incremented by *1/cwnd* each time an ACK is received. This is a linear growth of *cwnd*. The increase in *cwnd* should be at most one segment each RTT regardless how many ACKs are received in that RTT. Note that SS increments *cwnd* by 1 each time an ACK is received. For each RTT round, *cwnd* is increased by the number of ACKs received in a RTT. That is an exponential growth.

2.2.2.3 *Fast Retransmit / Fast Recovery*

Fast retransmit algorithm is adopted for the TCP sender to detect and repair loss, based on incoming duplicate ACKs.
When an out-of-order segment arrives, the TCP receiver sends a duplicate ACK immediately. This ACK indicates that a segment was received out-of-order and informs which sequence number is expected. The fast retransmit algorithm uses the triple duplicated ACKs (4 identical ACKs without the arrival of any other intervening packets) as an indication that a segment has been lost. In this case, TCP retransmits the segment that appears to be lost without waiting for the retransmission timer to expire. After that, the “fast recovery” algorithm governs the transmission of new data until a non-duplicate ACK arrives.
The “fast recovery” algorithm is usually implemented together with the “fast retransmit” algorithm, the motivation is to prevent the receiver’s buffer from emptying after fast retransmit. It is realized as follows:

- According to [16] and [17], when the triple duplicated ACKs are received, set *ssthresh* to
$$\max\left(\frac{\min(\mathrm{cwnd}, \mathrm{rwnd})}{2}, 2\right).$$

- Retransmit the lost segments and set *cwnd* to *ssthresh* plus 3 data segments. This artificially inflates the congestion window by the number of segments (three) that have left the network and which the receiver has buffered.

- For each additional duplicate ACK received, increment *cwnd* by a segment size. This artificially inflates the congestion window in order to reflect the additional segment that has left the network.
- Transmit a segment, if allowed by the new value of *cwnd* and the receiver's advertised window
- When the next ACK arrives that acknowledges new data, set *cwnd* to *ssthresh* (the value set in step 1). This is termed "deflating" the window.

2.3 Retransmission Time Out

During data transmission, in the case that packets (or ACKs) are lost, and less than three duplicate ACKs are received, the sender will wait for a period of time (retransmission timeout *RTO*) and then resend the lost packets. What the approximate value of RTO should be is a delicate question. If the RTO is longer than necessary, that would result in longer delay for applications in the case of frequent losses. If the RTO is too small, that may result in premature retransmissions causing waste of communication resources such as bandwidth and processing time. In order to accommodate various delays, such as transmission delay, link propagation, the header procession time, the ACK generation time etc, the RTO should be at least greater than the RTT. In a dynamic environment, the actual RTT may vary over time. Thus RTO must be constantly updated by every RTT. In TCP the Jacobson/Karn’s algorithm is used for computing the RTO and is discussed in [18].

2.3.1 Jacobson’s algorithm

Jacobson's algorithm is used to estimate the round trip time (EstRTT) and incorporates a simple measure of the variance. It works as follows.

Define:

`EstRTT`	estimated RTT
`SRTT`	sample RTT. That is the time from the moment a TCP packet is transmitted until an ACK is received for that packet
`SERR`	error between EstRTT and SRTT
`SDEV`	standard deviation

Then the RTO is obtained from the following equations:

$$EstRTT_{k+1} = (1-\alpha)\times EstRTT_k + \alpha \times SRTT_{k+1} \ , \tag{2.1}$$

$$SERR_{k+1} = SRTT_{k+1} - EstRTT_k \ , \tag{2.2}$$

$$SDEV_{k+1} = (1-\beta)\times SDEV_k + \beta \times \left|SERR_{k+1}\right| \ , \tag{2.3}$$

$$RTO_{k+1} = EstRTT_{k+1} + 4 \times SDEV_{k+1} \, . \tag{2.4}$$

According to [18] the appropriate initial values for *EstRTT* and *SDEV* (in seconds) satisfy the following equations:

$$EstRTT = SRTT \ , \tag{2.5}$$

$$SDEV = SRTT / 2 \, . \tag{2.6}$$

A typical value used for α is 0.125 and 0.25 for β.

For the simple case where all *SRTT*s are constant and equal to a fixed value RTT, then we can find the follow expression for *RTO(N)* from equations (2.1) – (2.6), where *N* represents the number of updates of the RTO preceding the occurrence of a time-out:

$$RTO(N) = RTT + 2RTT\left(\frac{3}{4}\right)^{N-1}. \tag{2.7}$$

2.3.2 *Karn's algorithm*

In Jacobson's algorithm, the SRTT must be measured to determine the EstRTT. But if a packet loss occurred, it is difficult to measure the SRTT correctly because it is not clear for the sender whether the received ACK is for the lost packet or for the retransmitted packet. To ensure those ambiguous SRTT will not corrupt the calculation of the EstRTT, Karn's algorithm is used for selecting the SRTT measurements. This algorithm works as follows.

- When an acknowledgement arrives for a packet that has been sent more than once, ignore any round-trip measurement result based on this packet.
- Calculate the backoff RTO when retransmission occurs. The RTO scaling factor is doubled to the value of the last after each retransmission, up to its maximum value of 64.
- Use the backoff RTO until an ACK arrives for the segment that has not been retransmitted.
- Only when the packet is acknowledged without an intervening retransmission will the RTO be recalculated from EstRTT.

Before the first SRTT measurement is selected, the standard initial value of RTO in the simulation package ns-2, see [9], is 6 seconds and in document RFC2988 [18] it is 3 seconds.

2.3.3 *The minimum RTO*

In document RFC2988 [18], the minimum RTO has been fixed to 1 second. But the RTTs commonly found in the Internet vary from 50 to 500 milliseconds. So it is not fair to use a fixed minimum RTO for both large RTTs and small RTTs. In addition, under a bad network connection, the probability that a RTO occurs for a larger RTT is larger than that for a smaller RTT if the RTO is fixed and independent of RTT. In [19], it is mentioned that the simple empirical heuristic of $RTO = 4RTT$ works reasonably well to provide fairness within TCP. In [20], the minimum RTO is defined as

$$RTO_{\min} = RTT_{\max} + 2 * G,$$

where RTT_{max} is the maximum of the two last RTT measurements and *G* is the granularity of the timer at the TCP sender. In [20], the granularity timer is set to $G = 10\,ms$. In [21] and [22], this setting has been shown to give good performance.

From (2.7), the maximum value of the RTO is derived as 3*RTT.* To reduce the complexity of determining the RTOs, mostly we take in our ns-2 simulations 4*RTT* as the minimum value of RTO so that in our models we can estimate the RTO by its minimum value. But to ensure the accuracy of the model, in some simulations we also use another minimum RTO value for testing the model.

2.4 New TCP standards and flavors

There are a couple of TCP implementations, including Tahoe, Reno, NewReno, SACK, and Vegas, due to the constant evolution of TCP, see [23]. Below we list the main characteristics of these versions,

- Tahoe is a relative early implementation. It contains Slow Start, Congestion Avoidance, and also the Fast Retransmit algorithm.
- Reno is an extension of the Tahoe version and it includes also Fast Recovery.
- NewReno redefined the fast retransmit/recovery algorithm. When TCP Reno performs fast retransmit, only one packet is retransmitted. So a problem can occur when multiple packets are dropped in a window of data. In NewReno, a partial ACK is taken as an indication of another lost packet and as such the sender retransmits the first unacknowledged packet. Unlike Reno, partial ACKs do not take NewReno out of fast recovery. This way, it retransmits one packet per RTT until all the lost packets are retransmitted and avoids requiring multiple fast retransmits from a single window of data.
- The selective acknowledgement (SACK) offers the possibility to the client to additionally report out-of-order data. This prevents TCP from unnecessarily resending certain segments.
- Vegas differs from the other TCP variants; it does not wait for loss to trigger *cwnd* reductions. If congestion exists in the network, the actual throughput is less than the expected throughput. This idea is used in Vegas to decide if it should adjust the window size.

In this report we will only concentrate on the Reno version of TCP because it contains the most commonly used features of Internet. Figure 2-3 gives an illustration of different working phases of TCP Reno.

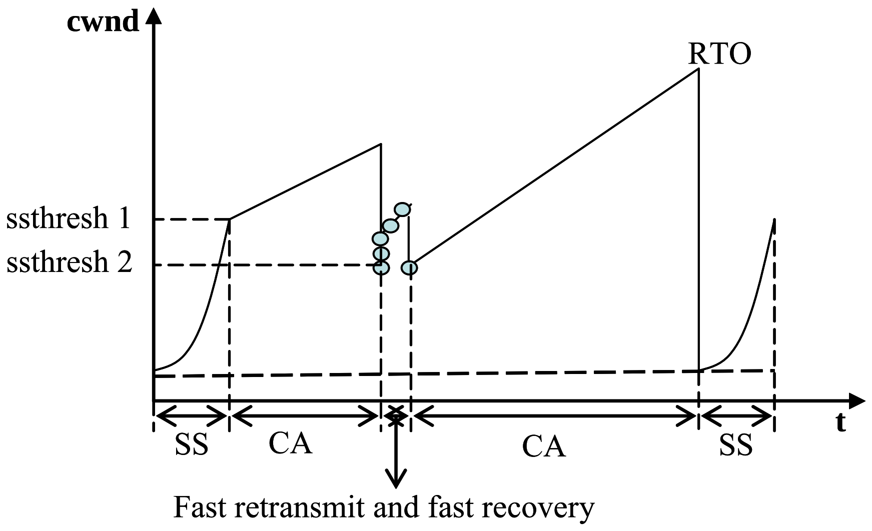


Figure 2-3: Diverse working phases in TCP connection.

The TCP description in this chapter contains the main characteristics of TCP, such that it is sufficient within the framework of our research; see [16], [12], and [24] for a more detailed description.

3 Modelling TCP's performance

In this chapter, we describe in Section 3.1 and Section 3.2 the model for predicting the mean TCP throughput by Mathis et al [6] and by Padhye et al [5] respectively. The Mathis model provides an upper bound on TCP's throughput. It focuses only on the TCP Congestion Avoidance (CA) mechanism. Unlike the Mathis model, the model of Padhye captures the behaviour of TCP's fast retransmit mechanism and also the effect of TCP's time-out events. However, both models ignore the effects of the SS and the fast recovery mechanism. Therefore, we extend the Padhye model by including those two effects. The extended model is described in Section 3.3. Finally the result of the comparison between the Padhye model and its extension and simulation results obtained from ns-2 are discussed in Section 3.4.

3.1 Model by Mathis

The simplest performance model for TCP, presented in [6], provides an upper bound on TCP's throughput - or simply put, TCP's average sending rate. This model is focused on the TCP Congestion Avoidance algorithm and assumed that a packet is dropped from a TCP connection if and only if the congestion window size has increased to a certain number of packets denoted as *W*. Another assumption is that the packet-loss process is periodic.
Applying the assumptions to the congestion avoidance algorithm, we get the following conclusions: the congestion window is halved when a packet is dropped and after the drop the TCP sender increases linearly its congestion window until the congestion window has reached its old value *W* and then another packet drop occurs. The time interval between two consecutive packet losses is referred to as a cycle. It is easy to verify that a cycle takes $W/2$ rounds, thus $W/2 \times RTT$ seconds. Figure 3-1 shows the illustration.

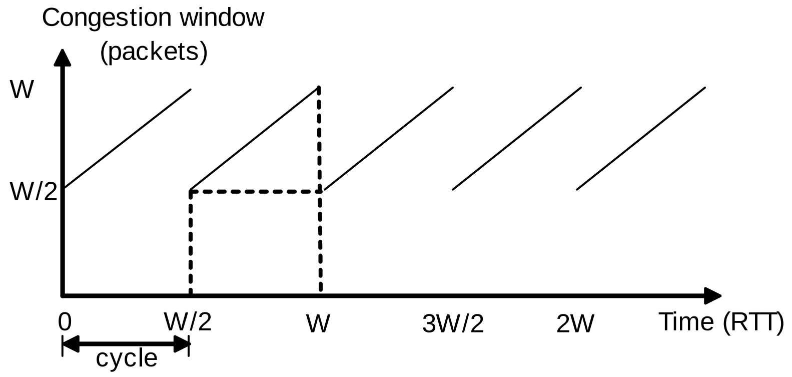


Figure 3-1: A periodic model of TCP window dynamics in steady state.

In every cycle, assuming packets of *B* bytes and a constant roundtrip time of RTT seconds, it is clear that the total data delivered during this cycle is

$$\left(\frac{W}{2}\right)^2 + \frac{1}{2}\left(\frac{W}{2}\right)^2 = \frac{3}{8}W^2 \text{ packets,} \tag{3.1}$$

and thus $B\left(\frac{3}{8}W^2\right)$ bytes.

Dividing the amount of data sent per cycle by the time it takes to send the data, we get the follow expression for the TCP's throughput, denoted by $\overline{X}$:

$$\overline{X} = \frac{data\ per\ cycle}{time\ per\ cycle} = \frac{B\left(\frac{3}{8}W^2\right)}{RTT\left(\frac{W}{2}\right)} = \frac{3B}{4RTT}W\ . \tag{3.2}$$

Because of the constant packet-loss probability of *p*, the expected number of consecutive packets delivered by the link between drops is $1/p$. That is also the total data delivered per cycle. So solving for *W* we get

$$W = \sqrt{\frac{8}{3p}}\ . \tag{3.3}$$

By substitution of (3.3) into (3.2), the expression of throughput is converted to

$$\overline{X}(p) = \frac{B}{RTT}\sqrt{\frac{3}{2p}}\ . \tag{3.4}$$

This model is not expected to be applicable under a number of situations where pure CA does not fully control TCP performance. For instance, if the data receiver is announcing too small a window, then TCP's performance is likely to be fully controlled by the receiver's window and not at all by the CA algorithm. Those situations reduce the performance relative to what is predicted by the model. Thus in many practical situations, (3.4) is used as an upper bound of the mean TCP throughput [6], i.e.,

$$\overline{X}(p) \le \frac{B}{RTT}\sqrt{\frac{3}{2p}}\ . \tag{3.5}$$

3.2 Model by Padhye

The current implementations of TCP include additional dynamics. For instance, there are limits on the window size enforced by the receiver, and it is observed that in TCP timeouts occur frequently. In this section, the more detailed model of TCP described by Padhye in [5] is presented, that incorporates the essential elements modelled in section 3.1 while adding more realistic aspects to represent the protocol more fully. This model focuses on CA behaviour of TCP and its impact on throughput, taking into account the dependence of CA on ACK behaviour, the manner in which packet loss is inferred(e.g., whether by duplicate ACK detection, or by timeout), limitation by *rwnd*, and average round trip time (RTT).

Firstly, the assumptions used in this model are enumerated below,

- A flow sent by a TCP sender contains unlimited amount of data to send
- Packet loss in one round is independent of packet loss in any other rounds
- Packet losses are correlated among the back-to-back transmissions within a round. That is, if a packet is lost, all remaining packets transmitted until the end of that round are also lost
- The time needed to send all the packets in a window is much smaller than the RTT.

In this section, we structure this model with features added gradually, that makes the model evolve in a clear manner.

3.2.1 Duplicate ACKs

As a first step, we assume that packet loss is detected exclusively by triple duplicate ACKs (TD). This loss detection results only in the current window size being halved. See Figure 3-2.

In this figure, we define:

TDP	a TD period, that is the period between packet losses indicated by TD
TDP_i	the ith TDP
W_{U_i}	the window size while the first packet loss occurs in the TDP_i, ignoring the window limitation
W_U	the window size when the first packet loss occurs in a TDP without the window limitation
A_i	the duration of TDP_i
A	the duration of a TDP
Y_i	the number of packets transferred in A_i
Y	the number of packets transmitted in a TDP

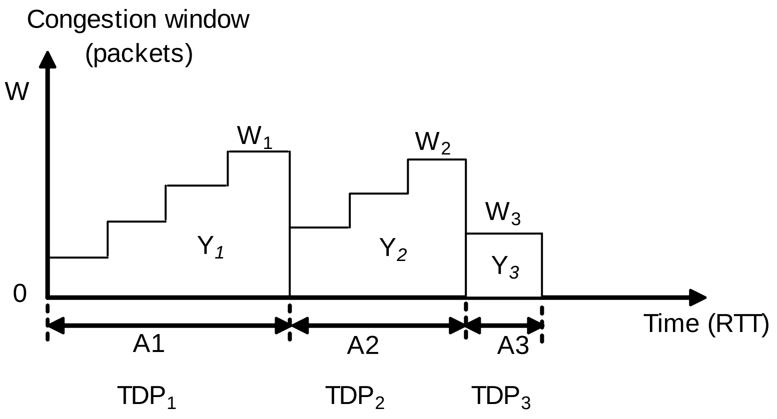


Figure 3-2: Evolution of window size over time when loss indications are TD.

We focus on one TDP period, represented by the symbol TDP_i. How the window size is regulated during TDP_i is presented in Figure 3-3. This figure is drawn in the unit of packets. The following conditions hold,

p	the probability that a packet is lost, given that the preceding packet in its round is not lost
$\overline{X}$	the mean TCP throughput
α_i	the number of packets sent in TDP_i up to and including the first lost packet
α	the number of packets sent in a TDP up to and including the first lost packet
β_i	the number of packets sent in the last round of the TDP_i
β	the number of packets sent in the last round of a TDP
r_{ij}	the duration of the jth round of TDP_i
r	the duration of a round in a TDP
X_i	the round number where the first loss occurs in TDP_i

X	the round number where the first loss occurs in a TDP
b	the number of packets that are acknowledged by a received ACK

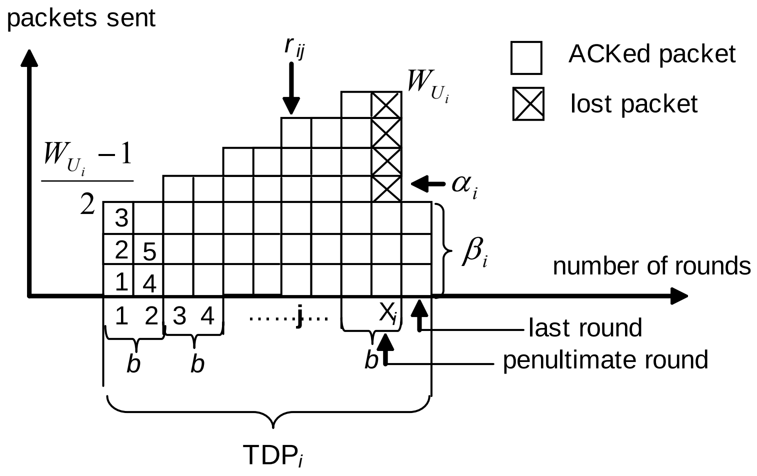


Figure 3-3: Packets sent during a TD period.

In the last round the amount of data sent is roughly equal to the number of packets acknowledged in the penultimate round. Therefore it is easy to verify that after α_i, $W_{U_i}-1$ more packets are sent before a TD loss indication occurs. Hence, the total number of packets sent in this period is $Y_i=\alpha_i+W_{U_i}-1$. Using this result, the expression for the expected number of packets transmitted in a TDP is

$$E[Y]=E[\alpha]+E[W_U]-1. \tag{3.6}$$

Because the packet loss probability is p, the expected number of consecutive packets delivered between drops is $1/p$. That is leading to,

$$E[\alpha]=\frac{1}{p}. \tag{3.7}$$

Substituting (3.7) into (3.6), we have

$$E[Y]=\frac{1-p}{p}+E[W_U]. \tag{3.8}$$

Because during the CA phase, at each round the window is incremented by $1/b$ and the number of packets sent per round is incremented by one every b rounds, the final window size W_{U_i} can be given by $W_{U_i}=W_{U_{i-1}}/2+X_i/b$. Using this result, the expression for the expected window size when the first packet loss occurs is derived:

$$E[W_U]=\frac{2}{b}E[X]. \tag{3.9}$$

Y_i can also be computed by counting the packets in each round. This leads to another expression for Y_i:

$$Y_i=\sum_{k=0}^{X_i/b-1}\left(\frac{W_{U_{i-1}}}{2}+k\right)b+\beta_i=\frac{X_i}{2}\left(\frac{W_{U_{i-1}}}{2}+W_{U_i}-1\right)+\beta_i. \tag{3.10}$$

Assume that the number of rounds in a period is independent of the window size. Then from the previous equation follows

$$E[Y]=\frac{E[X]}{2}\left(\frac{E[W_U]}{2}+E[W_U]-1\right)+E[\beta]. \tag{3.11}$$

Assuming β is uniformly distributed between 1 and $W_{U_i}-1$, we have

$$E[\beta]=\frac{E[W_U]}{2}. \tag{3.12}$$

From (3.8), (3.11), and (3.12), $E[W_U]$ can be determined as follows:

$$E[W_U]=\frac{2+b}{3b}+\sqrt{\frac{8(1-p)}{3bp}+\left(\frac{2+b}{3b}\right)^2}\ . \tag{3.13}$$

Substituting (3.13) into (3.9) and solving for *E[X],* it follows

$$E[X]=\frac{2+b}{6}+\sqrt{\frac{2b(1-p)}{3p}+\left(\frac{2+b}{6}\right)^2}\ . \tag{3.14}$$

Using (3.14), the expected length of the TDP can be derived in the following steps.
At the beginning, we have the assumption that the round trip times r_{ij} are independent of the window size, thus they are also independent of the round number *j*. Because the duration of TDP_i can be expressed as

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

the expected length of the TDP, denoted by *A*, can be expressed as:

$$E[A]=(E[X]+1)E[r], \tag{3.15}$$

where $E[r]=RTT$, and the expression of *E[X]* can be found in (3.14).
The throughput $\overline{X}(p)$ is defined as

$$\overline{X}(p)=\frac{E[Y]}{E[A]}.$$

Via equations (3.9), (3.11), (3.13) and (3.15), we have

$$\overline{X}(p)=\frac{E[Y]}{E[A]}=\frac{\frac{1-p}{p}+E[W_U]}{E[A]}=\frac{\frac{1-p}{p}+\frac{2+b}{3b}+\sqrt{\frac{8(1-p)}{3bp}+\left(\frac{2+b}{3b}\right)^2}}{RTT\left(\frac{2+b}{6}+\sqrt{\frac{2b(1-p)}{3p}+\left(\frac{2+b}{6}\right)^2}+1\right)}. \tag{3.16}$$

3.2.2 Loss indications including ACKs and time outs

In this section, the model is extended to include time-out loss indications. This occurs when packets (or ACKs) are lost, and less than three duplicate ACKs are received. In this case, the sender waits for the ACKs until the *RTO* expires and then retransmits the non-acknowledged packets. Following a time-out event, the *cwnd* is reset to one. In the case time-out occurs consequently, the Karn's back-off algorithm is applied. Figure 3-4 illustrates this.

First, the new variables introduced in this section are defined.

Z_i^{TO}	the duration of a sequence of time-outs
Z_i^{TD}	the time interval between two consecutive time-out sequences.
n_i	the number of TD periods in interval Z_i^{TD}
Y_{ij}	the number of packets sent in the *j*-th TD period of interval Z_i^{TD}
A_{ij}	the duration of the *j*-th TD period of interval Z_i^{TD}
X_{ij}	the number of the rounds in the *j*-th TD period of interval Z_i^{TD}
W_{ij}	the window size at the end of the *j*-th TD period of interval Z_i^{TD}
S_i	the sum of Z_i^{TO} and Z_i^{TD}
M_i	the number of packets sent during S_i
R_i	the total number of packet transmissions in Z_i^{TO}
Q	probability that the loss indication is a time-out

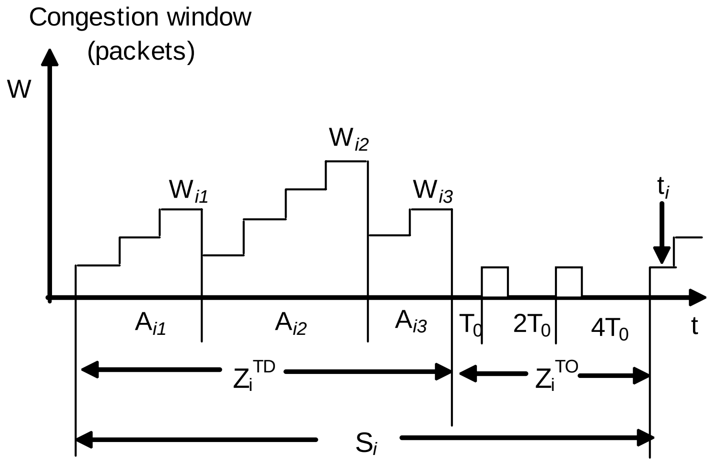


Figure 3-4: Evolution of window size when loss indications are TDs and TO.

According to the definitions above, the throughput for this section can be revised to

$$\overline{X}(p) = \frac{E[M]}{E[S]} . \tag{3.17}$$

Under the assumption that $\{n_i\}_i$ are i.i.d. (identically and independently distributed) random variables, independent of $\{Y_{ij}\}$ and $\{A_{ij}\}$, then $E[M]$ and $E[S]$ can be expressed as follows:

$$E[M] = E[n]E[Y] + E[R], \tag{3.18}$$

$$E[S] = E[n]E[A] + E[Z^{TO}] . \tag{3.19}$$

Because only one time-out happens during each Z_i^{TD}, the probability that a packet loss is indicated by time-out is given by $Q = 1/E[n]$, therefore (3.17) can be adapted to

$$\overline{X} = \frac{E[Y] + Q * E[R]}{E[A] + Q * E[Z^{TO}]} \tag{3.20}$$

where

$$E[R] = \sum_{k=1}^{\infty} kp^{k-1}(1-p) = \frac{1}{1-p}. \tag{3.21}$$

$E[Z^{TO}]$ is worked out based on the Karn's back-off algorithm. For the details about determining $E[Z^{TO}]$ and computing Q, see [5]. Here we just give the conclusions:

$$E[Z^{TO}] = T_0 \frac{f(p)}{1-p}, \tag{3.22}$$

where

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

and T_0 denotes the average duration of the first time out in a sequence of one or more successive time outs. Q is computed as follows:

$$Q(w) = \min\left(1, \frac{\left(1-(1-p)^3\right)\left(1+(1-p)^3\left(1-(1-p)^{w-3}\right)\right)}{1-(1-p)^w}\right), \tag{3.23}$$

where w represents the window size. Substituting (3.8), (3.15), (3.21), (3.22), and (3.23) into (3.20), we obtain

$$\overline{X}(p) = \frac{\frac{1-p}{p} + E[W_U] + Q(E[W_U])\frac{1}{1-p}}{RTT(E[X]+1) + Q(E[W_U])T_0 \frac{f(p)}{1-p}}. \tag{3.24}$$

3.2.3 *The impact of window limitation*

At the beginning of TCP flow establishment, the receiver advertises a maximum buffer size which determines a maximum congestion window size, W_{max}. As a consequence, during a period without loss indications, the window size can grow up to W_{max}, and then remains in that situation. See Figure 3-5.

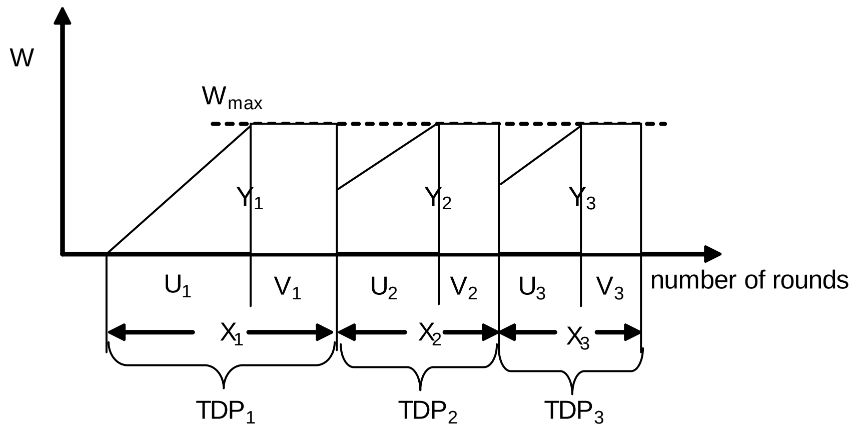


Figure 3-5: Evolution of window size when limited by W_{max}.

Define:

W	the expected window size when loss occurs
U_i	the duration to increasing the *cwnd* to W_{max} in period TDP_i
U	the duration to increasing the *cwnd* to W_{max} in a *TDP*
V_i	the duration between the moment that the *cwnd* reaches W_{max} and the moment the loss occurs in period TDP_i
V	the duration between the moment that the *cwnd* reaches W_{max} and the moment the loss occurs in a *TDP*

Because a window limitation exists, the window size cannot exceed W_{max}. We express W into two segments, that is

$$E[W] = \begin{cases} E[W_U] & \text{if } E[W_U] < W_{\max} \\ W_{\max} & \text{otherwise} \end{cases} \tag{3.25}$$

where $E[W_U]$ is given in (3.13). In the case of $\mathrm{E}[\mathrm{W_U}] \geq \mathrm{W_{max}}$, it is easy to verify that

$$E[U] = \frac{b}{2} W_{\max} . \tag{3.26}$$

Considering the number of packets sent in the TDP_i, we have

$$Y_i = \frac{U_i}{2}\left(\frac{W_{\max}}{2} + W_{\max}\right) + V_i W_{\max} . \tag{3.27}$$

This expression results in:

$$E[Y] = \frac{3}{4} W_{\max} E[U] + W_{\max} E[V] = \frac{3b}{8} W_{\max}^2 + W_{\max} E[V]. \tag{3.28}$$

From (3.8) and (3.28), $E[V]$ can be solved:

$$E[V] = \frac{1-p}{pW_{\max}} + 1 - \frac{3b}{8} W_{\max} . \tag{3.29}$$

Obviously, according to the definitions of U_i and V_i, we have

$$E[X] = E[U] + E[V] = \frac{b}{8} W_{\max} + \frac{1-p}{pW_{\max}} + 1 . \tag{3.30}$$

Plugging (3.30) into (3.24), we obtain the follow expression for the TCP throughput under the constraint of the window limitation:

$$\overline{X}(p) = \frac{\frac{1-p}{p} + W_{\max} + Q(W_{\max})\frac{1}{1-p}}{RTT\left(\frac{b}{8} W_{\max} + \frac{1-p}{pW_{\max}} + 2\right) + Q(W_{\max}) T_0 \frac{f(p)}{1-p}} . \tag{3.31}$$

Combining (3.24) and (3.31), the expression for TCP throughput $\overline{X}(p)$ becomes

$$\overline{X}(p)=\begin{cases}\dfrac{\dfrac{1-p}{p}+E[W]+Q(E[W])\dfrac{1}{1-p}}{RTT\left(\dfrac{b}{2}E[W]+1\right)+Q(E[W])T_0\dfrac{f(p)}{1-p}} & if\ E[W_u]<W_{\max} \\ \\ \dfrac{\dfrac{1-p}{p}+W_{\max}+Q(W_{\max})\dfrac{1}{1-p}}{RTT\left(\dfrac{b}{8}W_{\max}+\dfrac{1-p}{pW_{\max}}+2\right)+Q(W_{\max})T_0\dfrac{f(p)}{1-p}} & otherwise\end{cases}. \tag{3.32}$$

In the case that $p = 0$, the throughput reaches its upper bound. Because of the window limitation, the upper bound of $\overline{X}(p)$ can be deduced as W_{max}/RTT.

3.3 Extended TCP throughput model

In Section 3.2, the model only captures TCP's CA and fast retransmit algorithm. In this section we extend that model by considering the effects of the SS and the fast recovery mechanism. These two extensions are discussed in Section 3.3.1 and 3.3.2 respectively. In Section 3.3.1, we use the equations computed by Cardwell [11] for determining the amounts of data packets sent in the SS phase and the duration of the SS.
At first, we redefine a cycle as the time interval between two SS phases. For simplification, we assume SS is not ended by a time-out event. This implies that the SS phase is changing to the CA phase either after a loss is observed by a TD or after the *cwnd* reaches the *ssthresh*. For illustration, see Figure 3-6. The value of *ssthresh* has been discussed in Chapter 2. Based on that, we approximate the expected *ssthresh* as follows:

$$E[ssthresh]=\max\left(2,\frac{E[W]}{2}\right), \tag{3.33}$$

where *E[W]* is given in (3.25).

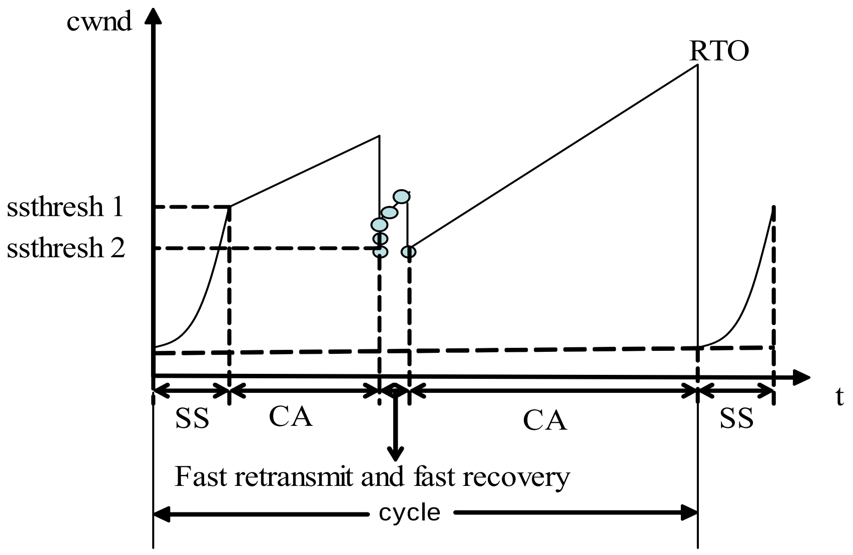


Figure 3-6: Description of a cycle.

3.3.1 Slow Start

As first step, we extend the model described in Section 3.2 by considering the SS phase. First new variables and functions used in this section are defined.

r	the exponential growth rate of the *cwnd* during a SS
w_1	the initial slow start window size
d_{ss}	the expected number of data packets sent during a SS
T_{ss}	the time taken to send d_{ss}, that is also the duration of a SS
W_{ss}	window size after sending d_{ss} packets
$W_{ss}(d)$	the window size achieved after sending *d* packets in the initial SS, where the window limitation is ignored
sendDataNr(w)	the number of data packets sent to increase the window size to *w* in the initial SS
$i(d)$	the number of rounds taken to transfer *d* data packets in the SS
$d_{ss}(p,d)$	the number of data segments expected to send in the SS before a loss occurs under the assumption that the total data segments size is *d* and the loss probability is *p*

According to [11], we get the following conclusions:

$$W_{ss}(d) = \frac{d(r-1)}{r} + \frac{w_1}{r}, \tag{3.34}$$

where

$$r = 1 + \frac{1}{b}.$$

$$i(d) = \begin{cases} \log_r\left(\dfrac{d(r-1)}{w_1} + 1\right) & \text{if } W_{ss}(d) \le W_{\max} \\ \log_r\left(\dfrac{W_{\max}}{w_1}\right) + 1 + \dfrac{1}{W_{\max}}\left(d_{ss}(p,d) - \dfrac{rW_{\max} - w_1}{r-1}\right) & \text{otherwise} \end{cases}, \tag{3.35}$$

$$d_{ss}(p,d) = \frac{\left(1-(1-p)^d\right)(1-p)}{d}. \tag{3.36}$$

Because the rounds number must be integer, we round (3.35) up. Thus (3.35) is modified as follows:

$$i(d) = \begin{cases} \left\lceil \log_r\left(\dfrac{d(r-1)}{w_1} + 1\right) \right\rceil & \text{if } W_{ss}(d) \le W_{\max} \\ \left\lceil \log_r\left(\dfrac{W_{\max}}{w_1}\right) + 1 + \dfrac{1}{W_{\max}}\left(d_{ss}(p,d) - \dfrac{rW_{\max} - w_1}{r-1}\right) \right\rceil & \text{otherwise} \end{cases}. \tag{3.37}$$

The function $\lceil n \rceil$ denotes the smallest integer larger or equal to n. From (3.34), the following equation for *sendDataNr(w)* can be derived:

$$sendDataNr(w) = \left(w - \frac{w_1}{r}\right)\frac{r}{r-1}. \tag{3.38}$$

Thus in order to let *w* reach *ssthresh*, the amount of data that needs to be sent is

$$sendDataNr(E[ssthresh]) = \left(E[ssthresh] - \frac{w_1}{r}\right)\frac{r}{r-1}. \tag{3.39}$$

This is also the maximum data amount that can be sent during the SS. For a packet loss probability larger than zero, this upper limit may not be reached. The expected amount of data sent in SS can be approximated as follows:

$$d_{ss} = d_{ss}(p, sendData(E[ssthresh])), \tag{3.40}$$

where the function $d_{ss}(.)$ is expressed in (3.36), and *sendData(E[ssthresh])* is given in (3.39). Because the duration of SS is the same as the time taken to send d_{ss}, it can be determined by

$$T_{ss} = RTT \times i(d_{ss}), \tag{3.41}$$

where function $i(.)$ is given in (3.37), and d_{ss} is computed in (3.40). According to (3.34), after sending d_{ss} data segments, *cwnd* is increased to

$$W_{ss} = W_{ss}(d_{ss}), \tag{3.42}$$

where function $W_{ss}(.)$ is given in (3.34), and d_{ss} is derived in (3.40).

3.3.2 Fast Recovery

When a packet loss is detected by TD, fast retransmission and fast recovery occur consequently. That can happen at the end of a SS or CA. We discuss these two situations separately. Before that, we take a close look at the fast recovery algorithm in order to find an expression for the expected data amount sent in this phase (denoted by d_{FR}), and the corresponding time it takes (T_{FR}).

According to the fast recovery algorithm, the value of *ssthresh* is halved to the window size when the loss occurs. Denote this original window size as *w*. After that, *cwnd* is reset to *ssthresh* plus 3, in the unit of packets. This procedure is summarized by the following equations:

$$ssthresh = \max\left(2, \frac{w}{2}\right), \tag{3.43}$$

$$cwnd = ssthresh + 3. \tag{3.44}$$

Under the condition that the lost packet is recovered successfully and that there is another ACK received for new data, fast recovery is ended and proceeds to CA. That implies, to terminate the fast recovery phase, at least two packets within *cwnd* must be sent successfully. Because the packet loss probability is *p*, the expected number of consecutive packets dropped by the link followed by a successful delivery is

$$\sum_{i=1}^{\infty} i \times p^{i-1}(1-p) = 1/(1-p). \tag{3.45}$$

Based on (3.44), the expected amount of data sent in the fast recovery state can be approximated as follows:

$$d_{FR} = cwnd - 2 + \frac{2}{1-p} = \max\left(2, \frac{w}{2}\right) + 1 + \frac{2}{1-p}. \tag{3.46}$$

In [11], the duration of fast recovery is assumed to be a single *RTT*. In our case, we take

$$T_{FR} = \frac{RTT}{1-p} \tag{3.47}$$

as the approximation of the time fast recovery lasts. This decision is based on the fact that every retransmission event of the same packet starts in a new round. Thus (3.45) can also be used as the approximation of the total number of rounds taken in the fast recovery phase.

3.3.2.1 *Fast Recovery at the end of SS*

In this section, we apply (3.46) and (3.47) to the case that fast recovery occurs at the end of SS. Let $d_{RecoverySS}$ denote the expected data amount sent during the fast recovery phase caused by loss indication in the SS. The corresponding duration is indicated by $T_{RecoverySS}$.

Firstly p_{ss}, denoting as the probability that the SS phase is ended by a loss detection, is determined. In Section 3.3.1, it is computed that the maximum number of data segments sent in SS is equal to *sendDataNr(E[ssthresh]),* thus p_{ss} can be expressed as follows:

$$p_{ss} = 1 - (1-p)^{sendDataNr(E[ssthresh])}. \tag{3.48}$$

This is also the probability of the occurrence of fast recovery at the end of SS. Therefore the expected number of data segments sent in this fast recovery phase is

$$d_{RecoverySS} = p_{ss} d_{FR} = p_{ss}\left[\max\left(2, \frac{w}{2}\right) + 1 + \frac{2}{1-p}\right]. \tag{3.49}$$

Since (3.42) demonstrates that the window size where loss occurs in SS is W_{ss}, w contained in (3.49) can be substituted by W_{ss}. Thus we have

$$d_{RecoverySS} = p_{ss}\left[\max\left(2, \frac{W_{ss}}{2}\right) + 1 + \frac{2}{1-p}\right]. \tag{3.50}$$

The time duration in this situation can be approximated as:

$$T_{RecoverySS} = p_{ss} T_{FR} = p_{ss}\frac{RTT}{1-p}. \tag{3.51}$$

3.3.2.2 *Fast Recovery in CA*

Assume there are *n* CA phases between two consecutive SS. Then the first $n-1$ CA phases are terminated by TD that result in fast recovery occurrences $n-1$ times. The last CA is ended by a time-out event. Therefore the number of data segments sent by fast recovery in CA is derived as follows.

$$d_{RecoveryCA} = (n-1)d_{FR} = (n-1)\left[\max\left(2, \frac{w}{2}\right) + 1 + \frac{2}{1-p}\right] \tag{3.52}$$

In Section 3.2.1, the average window size where the loss occurs at the end of CA is computed and given by *E[W]* in (3.25). So, substituting *w* contained in (3.52) by *E[W],* we obtain

$$d_{RecoveryCA} = (n-1)d_{FR} = (n-1)\left[\max\left(2, \frac{E[W]}{2}\right) + 1 + \frac{2}{1-p}\right]. \tag{3.53}$$

The time taken in this situation is equal to

$$T_{RecoveryCA} = (n-1)T_{FR} = (n-1)\frac{RTT}{1-p}. \tag{3.54}$$

As the next step, *n* must be determined. It is easy to verify that

$$Q(E[W]) = \begin{cases} \dfrac{1}{n+1} & \textit{if a loss happened in SS} \\ \\ \dfrac{1}{n} & \textit{otherwise} \end{cases}, \tag{3.55}$$

where the function *Q(.)* is given in (3.23). Because the loss probability in SS is equal to p_{ss}, (3.55) is modified as

$$Q(E[W]) = p_{ss}\frac{1}{n+1} + (1-p_{ss})\frac{1}{n}. \tag{3.56}$$

Solving *n* from the equation above, we get

$$n = \frac{1}{2}\frac{-Q(E[W]) + 1 + \sqrt{Q(E[W])^2 + 2Q(E[W]) + 1 - 4Q(E[W])p_{ss}}}{Q(E[W])}. \tag{3.57}$$

3.3.2.3 *Complete model*

Firstly new variables used in this section are defined.

d_{CA}	the total amount of data sent in CA phases
T_{CA}	the mean duration of CA phase
d_{T_0}	the total number of packet transmissions in a time-out sequence
T_{T_0}	the mean duration of a time-out sequence

According to Section 3.2, the variables that are defined above can be expressed as follows:

$$d_{CA} = nE[Y] = n\left(\frac{1-p}{p} + E[W]\right), \tag{3.58}$$

where *E[W]* is expressed in (3.25)

$$T_{CA} = nE[A] = \begin{cases} n \times RTT \times \left(\dfrac{b}{2}E[W] + 1\right) & \textit{if } E[W_u] < W_{\max} \\ n \times RTT \times \left(\dfrac{b}{8}E[W] + \dfrac{1-p}{pE[W]} + 2\right) & \textit{otherwise} \end{cases}, \tag{3.59}$$

$$d_{T_0} = E[R] = \frac{1}{1-p}, \tag{3.60}$$

$$T_{T_0} = E\left[Z^{T_0}\right] = \frac{f(p)}{1-p} T_0 , \tag{3.61}$$

where

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

Then the TCP throughput including the impacts of SS and fast recovery is expressed as follows:

$$\overline{X}(p) = \frac{\textit{total number sent in a cycle}}{\textit{duration of a cycle}} = \frac{d_{ss} + d_{\mathrm{Re}\,\mathrm{cov}\,erySS} + d_{CA} + d_{\mathrm{Re}\,\mathrm{cov}\,eryCA} + d_{T_O}}{T_{ss} + T_{\mathrm{Re}\,\mathrm{cov}\,erySS} + T_{CA} + T_{\mathrm{Re}\,\mathrm{cov}\,eryCA} + T_{T_O}} . \tag{3.62}$$

where dss, dRecoverySS, dRecoveryCA, Tss, TRecoverySS, and TRecoveryCA are given in (3.40), (3.49), (3.52), (3.41), (3.50), and (3.53), respectively, and where d_{CA}, T_{CA}, d_{T_0} , and T_{T_0} are expressed in (3.58)-(3.61) respectively.

In the case that $p = 0$ and the file size is finite, it is expected in [11] that all data packets are sent in the slow start phase. Then the expression below can be used to approximate the throughput in that situation.

$$\overline{X}(p) = \frac{d}{i(d) \times RTT} , \tag{3.63}$$

where *d* stands for the total data packet number and $i(d)$ is explained in Section 3.3.5.

3.4 Numerical results

The model of Padhye described in Section 3.2 and its extension are validated in this section by ns-2 simulations. In these simulations, TCP Reno flows are used. For simulating packet losses, the error model is added on the link between the router and the server. The error model simulates link-level errors or loss by either marking the packet's error flag or dumping the packet to a drop target. The error model contains different types of error modules. The error module we used here is ErrorModel/Periodic. The details of the error model are described in [9].

The following parameter settings are used for experiments:

- Bottleneck-link capacity: 200Mb/s
- Access-link capacity: 30Mb/s
- Mean file size: 1000 packets
- Packet size: 1000 bytes
- Maximum advertised window size: {8, 32}
- Fixed round-trip time: {0.06, 0.1, 0.2, 0.3, 0.5} seconds
- Loss probability: {0, 0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, 0.1}
- Minimum RTO: 4 times the fixed round-trip time

The simulation results are obtained by performing 10 independent simulation runs in which 100 servers are active to send the file with 1000 packets. The simulation topology is illustrated in Figure 3-7. In each run the mean throughput of 100 servers is obtained and finally the average throughput of the 10 individual runs is taken for comparing the throughput determined by the above-mentioned models. The comparison outcomes are also provided in Table 3-1. To make the difference between the model and the simulation clear, the relative error of the model is computed. The result is appended in the column ε_{rel} in Table 3-1.

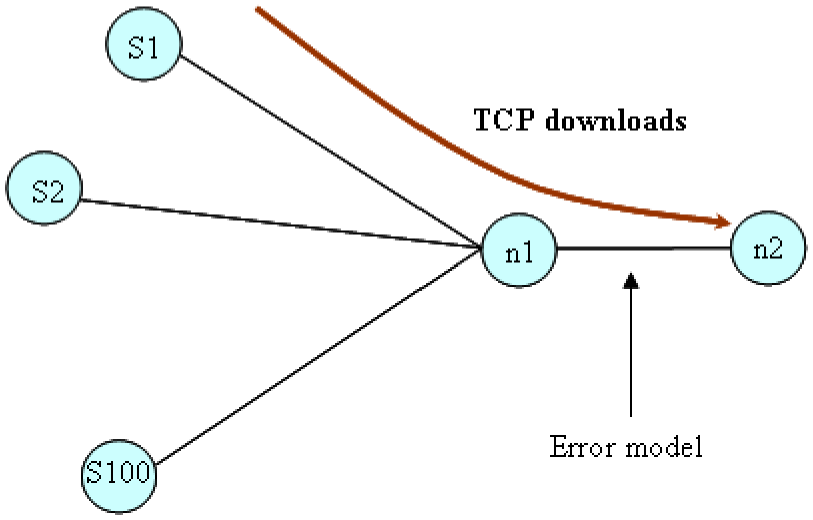


Figure 3-7: Simulation topology

	RTT = 0.06 s, Wmax = 8					RTT = 0.06 s, Wmax = 32				
p	NS	Model Padhye		Extended model		NS	Model Padhye		Extended model	
	throughput	throughput	ε_{rel}	throughput	ε_{rel}	throughput	throughput	ε_{rel}	throughput	ε_{rel}
0	128.62	133.33	**4%**	124.96	**-3%**	448.04	533.33	**19%**	450.29	**1%**
0.01	105.63	100.34	**-5%**	96.17	**-9%**	172.30	178.79	**4%**	167.35	**-3%**
0.02	87.48	85.47	**-2%**	80.85	**-8%**	113.06	119.15	**5%**	110.79	**-2%**
0.03	72.90	74.65	**2%**	70.32	**-4%**	84.78	90.89	**7%**	82.88	**-2%**
0.04	61.34	65.04	**6%**	61.15	**0%**	68.79	74.80	**9%**	70.38	**2%**
0.05	54.45	59.25	**9%**	55.66	**2%**	58.25	63.19	**8%**	59.00	**1%**
0.06	48.79	56.44	**16%**	52.42	**7%**	50.58	54.24	**7%**	50.24	**-1%**
0.07	42.53	48.01	**13%**	44.19	**4%**	43.98	48.17	**10%**	44.36	**1%**
0.08	38.65	42.12	**9%**	38.50	**0%**	39.53	42.82	**8%**	39.19	**-1%**
0.09	34.54	38.70	**12%**	35.26	**2%**	35.73	38.51	**8%**	35.05	**-2%**
0.1	30.84	34.90	**13%**	31.62	**3%**	32.30	34.97	**8%**	31.66	**-2%**
	RTT = 0.1 s, Wmax = 8					RTT = 0.1 s, Wmax = 32				
p	NS	Model Padhye		Extended model		NS	Model Padhye		Extended model	
	throughput	throughput	ε_{rel}	throughput	ε_{rel}	throughput	throughput	ε_{rel}	throughput	ε_{rel}
0	77.34	80	**3%**	75.92	**-2%**	269.51	320	**19%**	274.08	**2%**
0.01	63.74	62.00	**-3%**	59.46	**-7%**	105.19	109.94	**5%**	102.96	**-2%**
0.02	52.64	51.66	**-2%**	48.87	**-7%**	68.45	72.20	**5%**	67.16	**-2%**
0.03	43.97	44.86	**2%**	42.27	**-4%**	51.21	55.09	**8%**	50.24	**-2%**
0.04	37.69	39.90	**6%**	37.54	**0%**	41.53	44.79	**8%**	42.15	**1%**
0.05	32.58	36.01	**11%**	33.86	**4%**	35.18	37.98	**8%**	35.47	**1%**
0.06	28.82	32.71	**13%**	30.31	**5%**	30.45	32.79	**8%**	30.39	**0%**
0.07	25.58	29.00	**13%**	26.72	**4%**	26.90	28.90	**7%**	26.61	**-1%**
0.08	23.02	25.91	**13%**	23.74	**3%**	23.93	25.83	**8%**	23.65	**-1%**
0.09	20.89	23.24	**11%**	21.17	**1%**	21.59	23.18	**7%**	21.12	**-2%**
0.1	18.81	21.07	**12%**	19.10	**2%**	19.49	21.16	**9%**	19.19	**-2%**

	RTT = 0.2 s, Wmax = 8					RTT = 0.2 s, Wmax = 32				
p	NS	Model Padhye		Extended model		NS	Model Padhye		Extended model	
	throughput	throughput	ε_{rel}	throughput	ε_{rel}	throughput	throughput	ε_{rel}	throughput	ε_{rel}
0	38.73	40	**3%**	38.62	**0%**	135.01	160	**19%**	137.22	**2%**
0.01	31.93	31.20	**-2%**	29.92	**-6%**	51.76	54.00	**4%**	51.40	**-1%**
0.02	26.53	26.05	**-2%**	24.65	**-7%**	34.21	35.92	**5%**	33.39	**-2%**
0.03	22.19	22.62	**2%**	21.31	**-4%**	25.75	27.59	**7%**	25.16	**-2%**
0.04	18.98	20.07	**6%**	18.90	**0%**	20.85	22.36	**7%**	21.04	**1%**
0.05	16.47	18.08	**10%**	17.01	**3%**	17.68	19.05	**8%**	17.80	**1%**
0.06	14.51	16.52	**14%**	15.33	**6%**	15.29	16.40	**7%**	15.21	**-1%**
0.07	12.94	14.56	**12%**	13.42	**4%**	13.52	14.55	**8%**	13.41	**-1%**
0.08	11.64	12.99	**12%**	11.90	**2%**	12.11	13.03	**8%**	11.95	**-1%**
0.09	10.51	11.67	**11%**	10.63	**1%**	10.90	11.69	**7%**	10.65	**-2%**
0.1	9.57	10.61	**11%**	9.63	**1%**	9.83	10.63	**8%**	9.64	**-2%**
	RTT = 0.3 s, Wmax = 8					RTT = 0.3 s, Wmax = 32				
p	NS	Model Padhye		Extended model		NS	Model Padhye		Extended model	
	throughput	throughput	ε_{rel}	throughput	ε_{rel}	throughput	throughput	ε_{rel}	throughput	ε_{rel}
0	25.84	26.67	**3%**	25.84	**0%**	90.07	106.67	**18%**	92.08	**2%**
0.01	21.32	20.80	**-2%**	19.94	**-6%**	34.92	36.42	**4%**	34.68	**-1%**
0.02	17.66	17.42	**-1%**	16.48	**-7%**	22.93	24.41	**6%**	22.71	**-1%**
0.03	14.86	15.13	**2%**	14.26	**-4%**	17.25	18.47	**7%**	16.84	**-2%**
0.04	12.65	13.45	**6%**	12.67	**0%**	14.09	15.13	**7%**	14.25	**1%**
0.05	11.04	12.08	**9%**	11.36	**3%**	11.81	12.69	**7%**	11.85	**0%**
0.06	9.65	11.00	**14%**	10.21	**6%**	10.26	10.98	**7%**	10.19	**-1%**
0.07	8.60	9.71	**13%**	8.95	**4%**	9.06	9.77	**8%**	9.01	**0%**
0.08	7.78	8.69	**12%**	7.97	**2%**	8.11	8.71	**7%**	7.98	**-1%**
0.09	7.05	7.82	**11%**	7.13	**1%**	7.22	7.76	**7%**	7.07	**-2%**
0.1	6.36	7.04	**11%**	6.38	**0%**	6.59	7.11	**8%**	6.46	**-2%**
	RTT = 0.5 s, Wmax = 8					RTT = 0.5 s, Wmax = 32				
p	NS	Model Padhye		Extended model		NS	Model Padhye		Extended model	
	throughput	throughput	ε_{rel}	throughput	ε_{rel}	throughput	throughput	ε_{rel}	throughput	ε_{rel}
0	15.51	16	**3%**	15.54	**0%**	54.07	64	**18%**	55.27	**2%**
0.01	12.77	12.48	**-2%**	11.96	**-6%**	21.02	21.99	**5%**	20.94	**0%**
0.02	10.65	10.48	**-2%**	9.91	**-7%**	13.76	14.60	**6%**	13.58	**-1%**
0.03	8.94	9.12	**2%**	8.60	**-4%**	10.39	11.15	**7%**	10.17	**-2%**
0.04	7.59	8.05	**6%**	7.58	**0%**	8.43	9.06	**7%**	8.53	**1%**
0.05	6.60	7.27	**10%**	6.84	**4%**	7.10	7.64	**8%**	7.14	**1%**
0.06	5.82	6.62	**14%**	6.15	**6%**	6.16	6.60	**7%**	6.12	**-1%**
0.07	5.20	5.84	**12%**	5.38	**4%**	5.43	5.85	**8%**	5.40	**-1%**
0.08	4.68	5.22	**12%**	4.79	**2%**	4.89	5.21	**7%**	4.78	**-2%**
0.09	4.20	4.67	**11%**	4.26	**1%**	4.38	4.69	**7%**	4.28	**-2%**
0.1	3.84	4.26	**11%**	3.87	**1%**	3.97	4.27	**8%**	3.88	**-2%**

Table 3-1: A comparison of the mean throughput (packets/second) depicted by Model Padhye, the extended model, and simulation results

Table 3-1 shows that in the case that window limitation has no impact ($W_{\max} = 32$ in our situation) the extended model is very accurate for predicting the mean TCP throughput. The relative error of this model is not higher than 2%. Comparing this to the model of Padhye, it performs much better for all different RTTs and all loss probabilities chosen. If the window limitation has impact ($W_{\max} = 8$ in our case), the extended model performs a little worse for a probability lower than 0.03, but for probabilities larger than 0.03, the extended model performs much better. For $p = 0$, the relative errors of the extended model are within ±2%. The reason why it is not equal to 0% is that some facts are not taken into account in the model, for instance, the time TCP needs to establish the connection.
Summarizing, we conclude that the extended model in general is more accurate than the Padhye model.

4 An analytical model for the Response Time

Apart from the effect of the TCP throughput, the combined effect of the Response Time (*RT*) and the Total Download Time (*TDT*) is the most significant factor that determines the perceived end-to-end browsing quality [25]. Therefore, it is necessary to build models for predicting the RT and the TDT and find out on what parameters the RT and the TDT depends so that by changing some of these parameters the performance can be optimized. These two models are discussed in this chapter and in the further chapters.
First we develop a model for the RT: the time it takes to establish a TCP connection and the additional time it takes to send the first packet containing data. From a user's point of view, the RT is simply the time from clicking on a url link until the first packet arrives and something appears on the screen. In Cardwell et al [11], there is a model for predicting the TCP connection establishment time. But we cannot use this model to predict the RT. The first reason is that the Cardwell TCP connection establishment model assumes that when a SYN/ACK packet is lost, the receiver retransmits this lost packet. But we think it is more realistic that the sender retransmits the SYN packet when it does not receive the SYN/ACK packet after the RTO. The second reason is that this model just focuses on the time taken to send *two* packets successfully. But the RT is determined by the duration of sending *four* packets successfully. Therefore, in our work, we develop a new analytical model for predicting the RT. The RT model is described in Section 4.1. The validation of the model is described in Section 4.2. In Section 4.3, the RT model is extended in a multi-domain environment.

4.1 An analytical model for the Response Time

In this section we model the Response Time as the time it takes to establish a TCP connection and the additional time it takes to send the first packet containing data. Because every TCP connection starts with a "three-way handshake", in which the client and server exchange initial sequence numbers, the Response Time is determined by the time it takes to send 4 packets successfully. Here the first three packets are related to the three-way handshake while the fourth packet contains the first data. Figure 4-1 shows an example where none of the sent packets are lost.

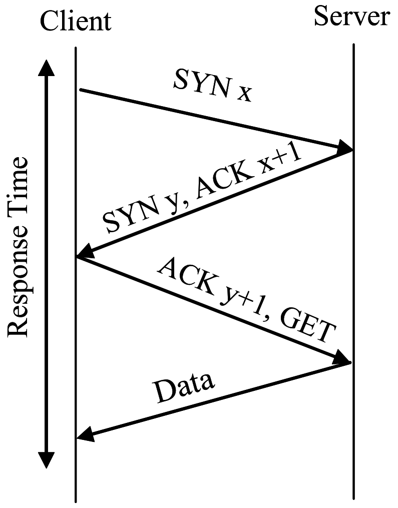


Figure 4-1: TCP connection establishment and first data packet

We will assume that the packet loss probabilities in forward and reverse direction are the same, which will be denoted by p . Suppose there are n failures (i.e., packet losses) before the first packet containing data is received. Failures can be classified as two kinds. The first kind is due to a packet loss from the client to the server. The second kind is caused by a packet loss from the server to the client. For both kinds of failures, packets will be retransmitted from the client side. See Figure 4-2.

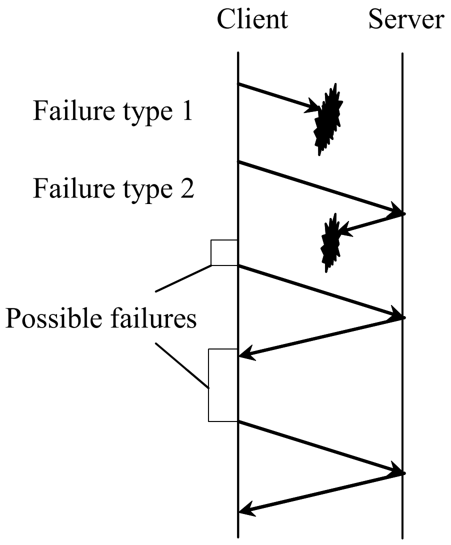


Figure 4-2: Receiving first packet containing data after one or more failures during transmission

We define the following parameters in our discussion:

P_n	probability of receiving the first data packet after exactly n packet losses
L_n	conditional mean response time when the first data packet is received after exactly n packet losses
L	the mean response time
j	the number of packet losses during transmission which belongs to *failure type 1*

We have,

$$P_n = \sum_{j=0}^{n} \binom{n}{j} p^j \left[p(1-p)\right]^{n-j} (n+1)(1-p)^2 (1-p)^2 . \tag{4.1}$$

Simplifying (4.1) in Maple [26], we obtain

$$P_n = (n+1)(1-p)^4 p^n (2-p)^n . \tag{4.2}$$

We set the packet loss probability $p = \{0.01, 0.05, 0.1, 0.25, 0.5\}$ to draw the probability P_n in Figure 4-3. It shows that for $n = 0$, lower p leads to higher P_n. This phenomenon implies that for a lower packet loss probability, the probability that the first data is send successfully without failures is higher than that for a higher packet loss probability. Another conclusion is that for a fixed p, P_n approaches to zero when n grows large enough. This conclusion implies in theory that the first data packet will be sent successfully sooner or later under the assumption that retransmission occurs when a packet is lost.

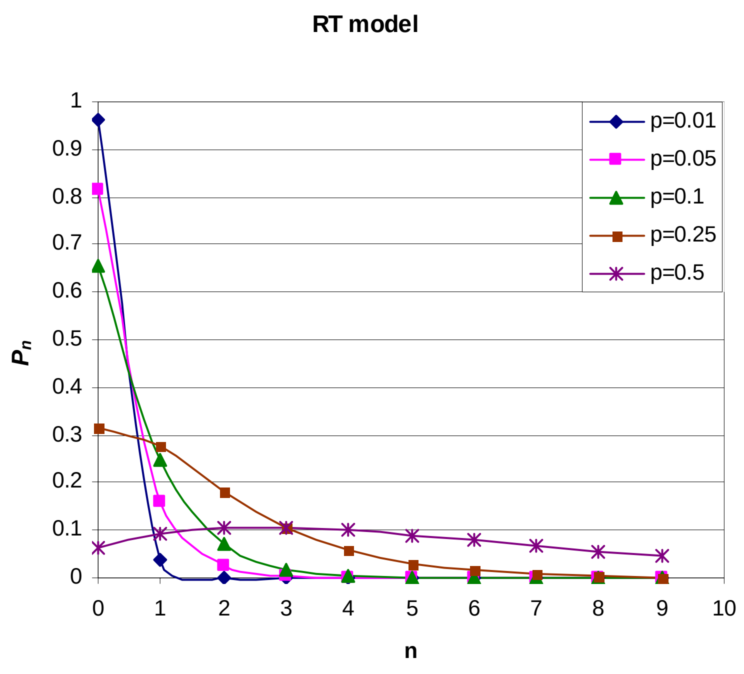


Figure 4-3: Probability of receiving the first data packet after n packet losses

At each stage drawn in Figure 4-2, loss is detected through a Retransmission Time Out (RTO). According to [18], upon its first update the Retransmission Timer becomes

$$T_1 = \max\{1, RTT + \max\{G, 2RTT\}\}, \tag{4.3}$$

where G denotes the TCP Timer Granularity. In many TCP implementations G is set to 500ms. In [11], it is shown that

$$L_n = \begin{cases} (2^n - 1)T + 2RTT; & n \le 6, \\ [63 + 64(n-6)]T + 2RTT; & n \ge 7, \end{cases} \tag{4.4}$$

where T satisfies $T = \frac{1}{n+1}T_1 + \frac{n}{n+1}T_0$.

Based on (4.2) and (4.4), after a tedious calculation, we can get an analytical expression for the mean response time L satisfying

$$L = \sum_{n=0}^{\infty} P_n L_n = \sum_{n=0}^{6} P_n L_n + \sum_{n=7}^{\infty} P_n L_n$$

$$L = \frac{A\left(b_0 + b_1 A + b_2 A^2 + b_3 A^3 + b_4 A^4 + b_5 A^5 + b_6 A^6 + b_7 A^7\right)}{1 - A} + 2RTT \tag{4.5}$$

where

$$A = p(2-p), b_0 = T_0 + T_1, b_1 = 3T_0, b_2 = 6T_0 + T_1$$

$$b_3 = 14T_0 + 2T_1, b_4 = 32T_0 + 4T_1$$

$$b_5 = 72T_0 + 8T_1, b_6 = 160T_0 + 16T_1, b_7 = -160T_0 - 32T_1.$$

The probability that the RT is t seconds or less is

$$P[L_n \le t] = \begin{cases} \sum_{n=0}^{N} p_n, & t \ge 2RTT \\ 0, & otherwise \end{cases}. \tag{4.6}$$

When $63\left(\frac{T_1 + 6T_0}{7}\right) + 2RTT \le t$, N must be obtained from the equation $[63 + 64(N-6)]T + 2RTT = t$. Otherwise N is the solution of the equation $(2^N - 1)T + 2RTT = t$. The outcome of N must be rounded down.

Under the condition $RTT = 0.4$ s and $T_0 = 6$ s, we set the packet loss probability $p = \{0.01, 0.05, 0.1, 0.25, 0.5\}$ to draw the probability $P[L_n \le t]$ in Figure 4-6. It shows the probability $P[L_n \le t]$ increases to 1 if t is long enough. With lower $p, P[L_n \le t]$ approaches to 1 faster. Those phenomena imply that the user sees something appear on the screen in theory sooner or later under the assumption that the loss probability is lower than 1. Another conclusion is that the probability that the user sees something on the screen within a fixed duration of time is higher for a lower loss packet probability.

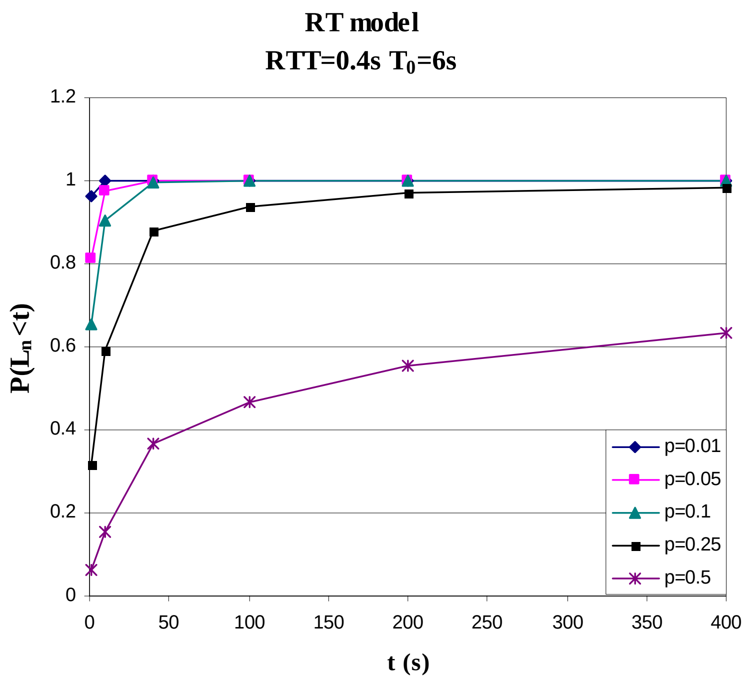


Figure 4-4: An example for cumulative distribution of RT

4.2 Numerical results of the RT model

In this section we report a simple ns-2 simulation experiment that has been run as a validation for the RT model. The topology of our simulation is depicted in Figure 4-5.

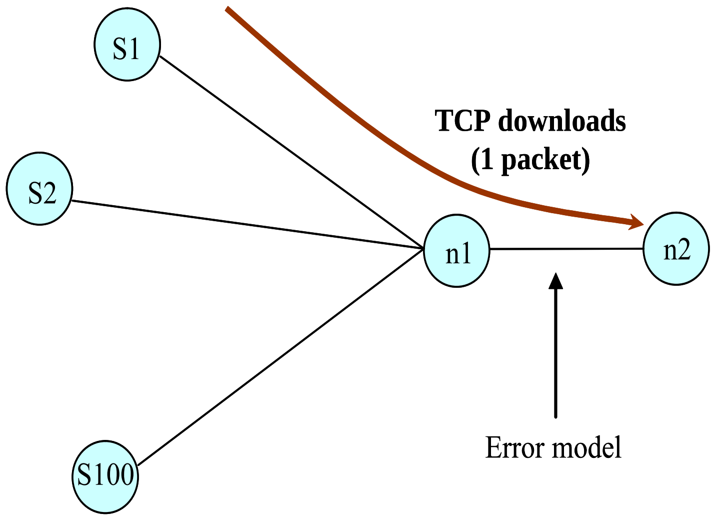


Figure 4-5: RT simulation topology

In this experiment, each file consists of 1 packet and the RTT is set to be 600 ms. The packet loss *p* at the aggregation link n1-n2 has been varied between 0-16%. It is assumed that the packet loss is random, i.e., without correlation between consecutive lost packets. In our simulation packets in both up- and downlink directions suffer from this packet loss. The ns-2 script has recorded actual values of the packet loss probability, RTT and the mean Response Time. The comparison between the model and the simulation is depicted in Figure 4-6.

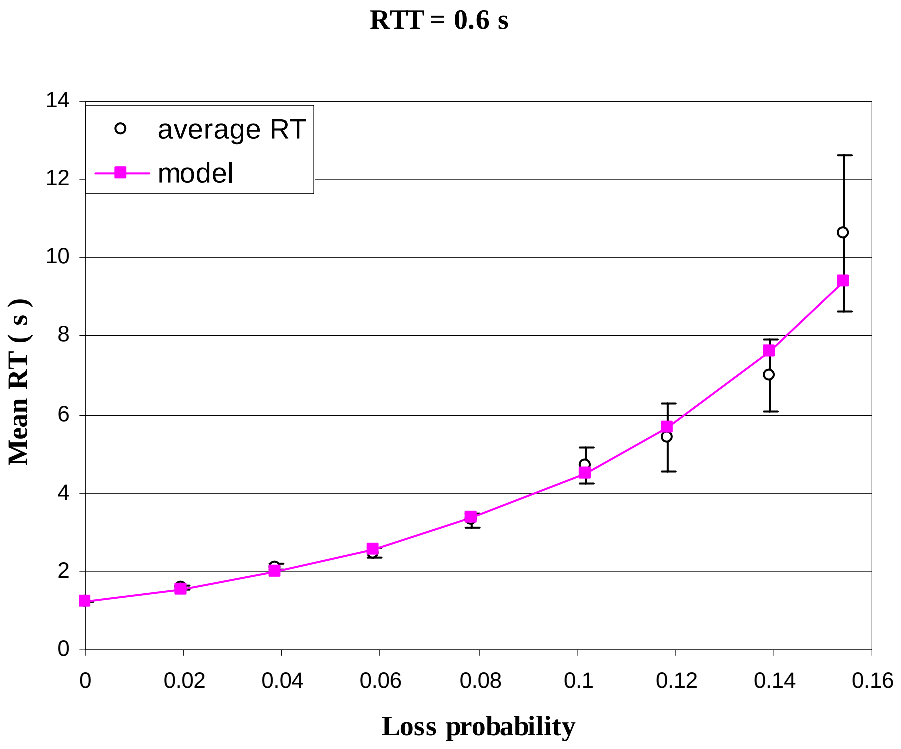


Figure 4-6: Comparison between model and simulation for mean Response Time in a single domain environment

We conclude from Figure 4-6 and various additional simulations that our model for the mean Response Time considered as a function of the loss probability is very accurate.

In fact, analytical results for the mean Response Time always fall within the 95% confidence interval of the simulated values.

4.3 Response Time in a multi-domain environment

In this section we present a model for the mean Response Time for TCP based downloads in the multi-domain setting. In Section 4.3.1 the proposed model is described and in Section 4.3.2 the numerical results are presented.

4.3.1 *The proposed model*

According to Section 4.2, the mean RT can be modelled as follows:

$$RT = f_1(T_0, RTT, p),$$

where the function $f_1(.)$ is given explicitly in (4.5).

In order to apply (4.5) in a multi-domain environment the network parameters *RTT* and *p* have to be determined, on the condition that these parameters are known for the individual domains. To be more precise, we assume that there are *N* network domains while for each of the network domains the following performance measures are specified between service provider and the network operators:

p_i :	packet loss probability for network *i*
RTT_i :	round trip time for network *i*

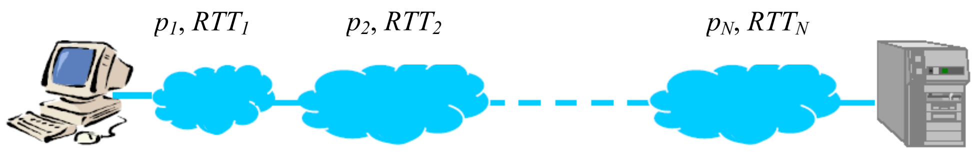


Figure 4-7: Packet loss probability (p_i) and round trip time (RTT$_i$) specifications per network

In our situation we will approximate the end-to-end performance measures parameters RTT and *p* from the packet loss probabilities and round trip times for the individual networks as follows:

$$RTT = \sum_{i=1}^{N} RTT_i \, , \tag{4.7}$$

and

$$p = 1 - \prod_{i=1}^{N} (1 - p_i) . \tag{4.8}$$

Those expressions combined with (4.5) give the desired model for Response Time for TCP based downloads in a multi-domain environment.

4.3.2 *Numerical results*

In this section we work out a concrete example of the use of the proposed model. We assume that a user crosses three domains in order to reach a server. See Figure 4-8.

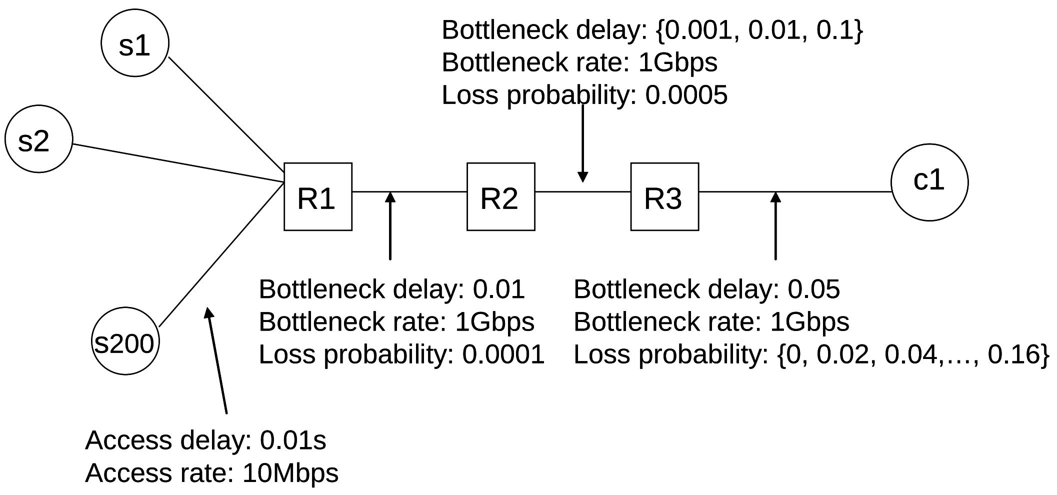


Figure 4-8: Simulation topology for RT model in a multi-domain environment

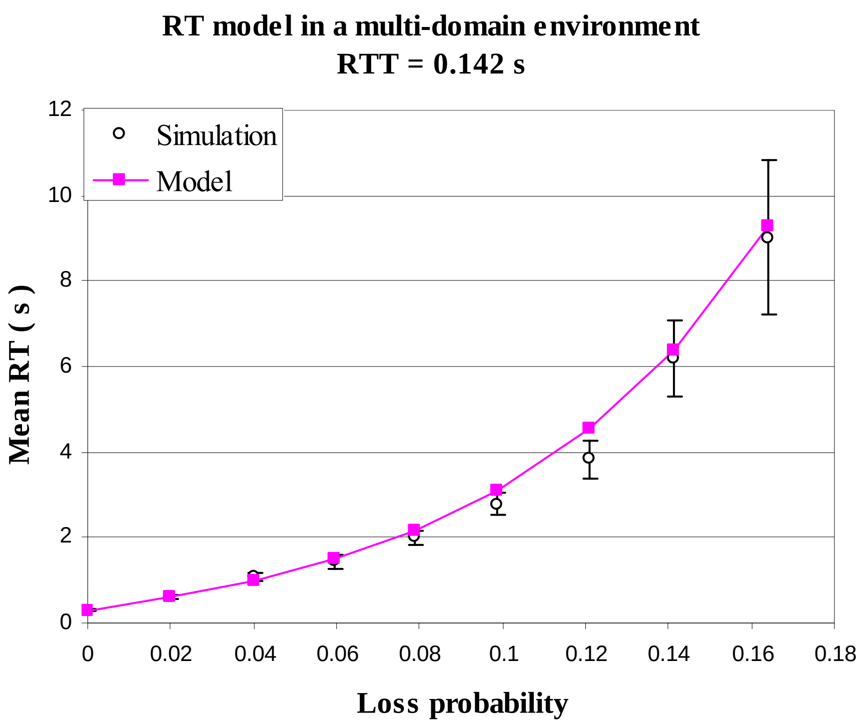
For the TCP parameters we choose the following values for both the simulations and the model:

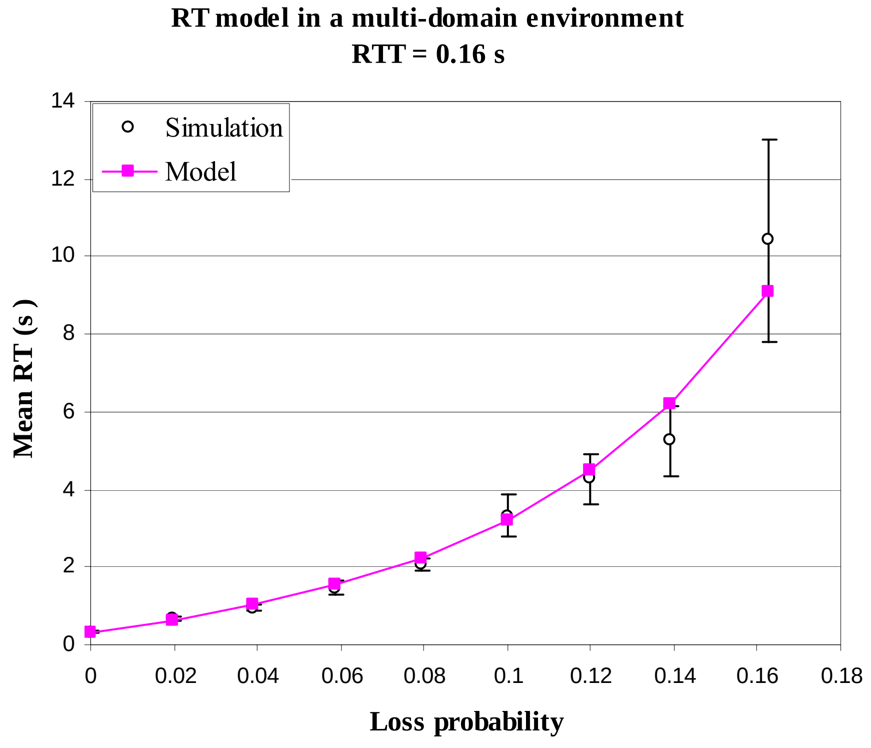
- MSS: 1640 Bytes
- The maximum receiver window size: 32 packets
- Minimum RTO: 1 s
- b: the number of packets that is acknowledged by each ACK: 1
- w_1: the initial window size: 1

In the four networks we have the following assumptions:

Network 1:	$p_1 : 0, RTT_1 : 0.02s, B_1 : 10Mbps$
Network 2:	$p_2 : 0.0001, RTT_2 : 0.02s, B_2 : 1Gbps$
Network 3:	$p_3 : 0.0005, RTT_3 : \{0.002, 0.02, 0.2\}s, B_3 : 1Gbps$
Network 4:	$p_4 : \{0, 0.02, ..., 0.16\}, RTT_4 : 0.1s, B_4 : 1Gbps$

Note that for each network the round trip time is two times the bottleneck delay.
Network 1 typically corresponds to a broadband access network with more bandwidth than ADSL, e.g., VDSL. Networks 2, 3, and 4 typically correspond to core networks.
The results are depicted in Figure 4-9.


RT model in a multi-domain environment
RTT = 0.142 s
Simulation
Model
Mean RT (s)
Loss probability
0
2
4
6
8
10
12
0
0.02
0.04
0.06
0.08
0.1
0.12
0.14
0.16
0.18


RT model in a multi-domain environment
RTT = 0.16 s
Simulation
Model
Mean RT (s)
Loss probability
0
2
4
6
8
10
12
14
0
0.02
0.04
0.06
0.08
0.1
0.12
0.14
0.16
0.18

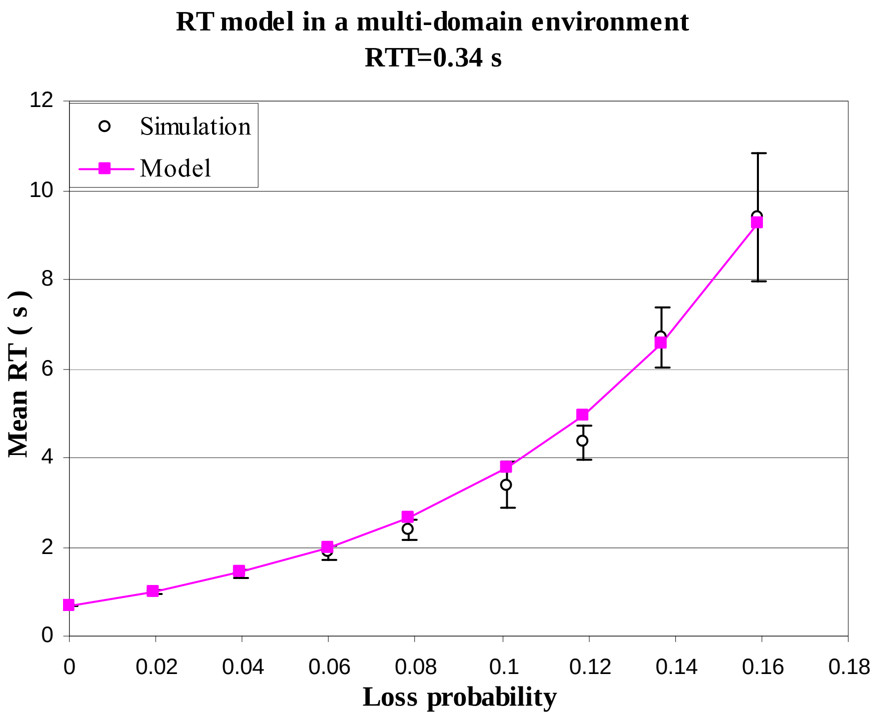


Figure 4-9: Comparison between model and simulation for the mean Response Time in a multi-domain environment

We conclude from Figure 4-9 that our model for the mean RT in a multi-domain environment is very accurate. In fact, analytical result for the mean RT in multi-domain environment mostly falls within the 95% confidence interval of the simulated values.

5 An analytical model for Total Download Time in a single-domain environment

In this chapter we describe an analytical model for computing the mean total data download time (TDT). TDT consists of a TCP connection establishment time and data transfer time (DTT), where DTT is the time between sending the first data packet and receiving the last data packet. From Section 2.2.1, we know the TCP connection establishment time is the time taken to send three packets successfully. Therefore, we approximate it by 3/4 times the *RT* that is discussed in Chapter 4. In this chapter, we focus on computing the DTT. In Cardwell et al [11], a DTT model is proposed under the assumption that packet loss happens only in the direction from sender to receiver. This model is described in Section 5.1 and validated in Section 5.2. But in the real situation, not only data segments can be lost during a TCP data transmission but also the ACKs of data packets can be dropped in the direction from the receiver to the sender. Therefore, we extend the Cardwell model by including the impact of the loss of ACKs. This extension is discussed in Section 5.3. The validation of the extended model is discussed in Section 5.4.

5.1 Cardwell model

Cardwell proposed a model that allows the computation of the DTT for files transported over TCP. In fact, in [11] the DTT is decomposed in four segments: the initial slow start phase, the possible delay from delayed acknowledgements, the delay due to the recovery of the first loss, and the transfer of the remaining data.

5.1.1 Initial Slow Start

The expected delay for the initial SS phase is denoted as $E[T_{ss}]$. It is determined under two distinguished situations. In one situation, the sender's window increases until the packet loss is observed which terminates the initial SS phase. Another situation is that the maximum window, denoted as $W_{\max}$, eventually bounds the sender's window until the packet loss is detected and the initial SS is ended.
In order to determine which case occurs, we need to compute $E[W_{ss}]$, that is the expected window size at the end of the SS phase by ignoring the window limitation. In the SS phase each time an ACK is received the congestion window is increased by one segment. Let *b* denote the number of packets that are acknowledged per ACK and let r denote the exponential growth rate of the window size during SS. Since one ACK is for every *b*-th data segment received, the total number of ACKs the sender receives by each round can be approximated as $cwnd / b$. It is deduced consequently,

$$r = 1 + \frac{1}{b}. \tag{5.1}$$

Assume a sender starts with an initial window of w_1 packets. Because the exponential growth rate of the window size is r, with the window size after the *i*-th round denoted by w_i, satisfies:

$$w_i = r^{i-1} w_1. \tag{5.2}$$

This is also the number of data packets sent during the *i*-th round. According to (5.2), $E[W_{ss}]$ is determined by the total number of rounds contained in the initial SS phase. This number depends further on $E[d_{ss}]$, denoting the expected number of data packets sent successfully during the initial SS. Hence we deduce $E[d_{ss}]$ first.

Denote the packet loss probability as *p* and the file size as *d*. Because the initial SS phase is ended only by loss indication, we have

$$E[d_{ss}] = d, \, if \;\; p = 0 \,. \tag{5.3}$$

That implies that all data packets are sent in the initial SS phase if no packet loss occurs. For $p > 0$, we have

$$E[d_{ss}] = \left(\sum_{k=0}^{d-1} (1-p)^k \times p \times k \right) + (1-p)^d \times d = \frac{\left(1-(1-p)^d\right)(1-p)}{p} \,. \tag{5.4}$$

Putting (5.1) and (5.2) together, we get the complete expression for $E[d_{ss}]$:

$$E[d_{ss}] = \begin{cases} d & if \;\; p = 0 \\ \dfrac{\left(1-(1-p)^d\right)(1-p)}{p} & otherwise \end{cases} \,. \tag{5.5}$$

Denote the number of rounds taken to send $E[d_{ss}]$ as *n*. Based on (5.2), we have the following equation:

$$w_1 + r \times w_1 + \cdots + r^{n-1} \times w_1 = E[d_{ss}] \,. \tag{5.6}$$

Solving *n* from (5.6), we have

$$n = \log_r \left(\frac{E[d_{ss}](r-1)}{w_1} + 1 \right). \tag{5.7}$$

According to (5.2) and (5.7), it can be shown that

$$E[W_{ss}] = \frac{E[d_{ss}](r-1)}{r} + \frac{w_1}{r} \,. \tag{5.8}$$

In fact, if $E[W_{ss}] \le W_{\max}$, then the window limitation has no impact during the data transmission and $E[T_{ss}]$ is simply the time to send $E[d_{ss}]$. It follows that

$$E[T_{ss}] = RTT \times n \;\; if \;\; E[W_{ss}] \le W_{\max} \,, \tag{5.9}$$

where n is given in (5.7).

If $E[W_{ss}] > W_{\max}$, then SS proceeds in two phases. First, *cwnd* grows up to $W_{max.}$ According to (5.2), this takes

$$i_1 = \log_r \left(\frac{W_{\max}}{w_1} \right) + 1 \text{ rounds.} \tag{5.10}$$

The number of data segments sent in these rounds is computed as follows:

$$d_1 = w_1 + r \times w_1 + \cdots + r^{i_1 - 1} \times w_1 = \frac{rW_{\max} - w_1}{r-1} \,. \tag{5.11}$$

When W_{max} is reached, the SS comes into the second phase. During this phase, the remaining data segments are sent at a rate of W_{max} packets per round. Then the number of corresponding rounds is

$$i_2 = \frac{E[d_{ss}] - d_1}{W_{\max}}. \qquad (5.12)$$

where $E[d_{ss}]$ and d_1 are given in (5.5) and (5.11), respectively. Therefore, the corresponding $E[T_{ss}]$ can be expressed as follows:

$$E[T_{ss}] = RTT(i_1 + i_2), \qquad (5.13)$$

where i_1 and i_2 are given in (5.10) and (5.12) respectively. Combining (5.9) and (5.13), we have the complete expression for $E[T_{ss}]$:

$$E[T_{ss}] = \begin{cases} RTT\left[\log_r\left(\frac{W_{\max}}{w_1}\right) + 1 + \frac{1}{W_{\max}}\left(E[d_{ss}] - \frac{rW_{\max} - w_1}{r-1}\right)\right] & if\ E[W_{ss}] > W_{\max} \\ RTT \times \log_r\left(\frac{E[d_{ss}](r-1)}{w_1} + 1\right) & if\ E[W_{ss}] \le W_{\max} \end{cases}.$$

(5.14)

5.1.2 *Delayed acknowledgements*

Delayed acknowledgements can cause relatively large delays for short transfers. This occurs only when delayed ACK's options are enabled. Then the receiver waits in vain for a second packet until finally the delayed ACK timer expires after which the receiver sends an ACK. Cardwell assumes in [11] that in UNIX platforms this delay is uniformly distributed between 0 ms and 200 ms while for Windows 95 and Windows NT 4.0 this delay is uniformly distributed between 100 ms and 200 ms, see [11].

Denoting the expected delay between the reception of a single segment and the delayed ACK for that segment as $E[T_{delack}]$ we obtain

$$E[T_{delack}] = \begin{cases} 0, & when\ b = 1\ or\ w_1 > 1 \\ 100ms, & when\ b > 1\ and\ w_1 = 1;\ Unix \\ 150ms, & when\ b > 1\ and\ w_1 = 1;\ Windows \end{cases}. \qquad (5.15)$$

5.1.3 *Delay due to recovery of the first loss*

Slow start ends with a packet loss if and only if a flow has at least one loss. The probability l_{ss} that this occurs satisfies

$$l_{ss} = 1 - (1-p)^d \qquad (5.16)$$

According to [5] the probability *Q(w)* that a sender detects a packet loss through a retransmission timeout (RTO), given that a packet loss occurs, is

$$Q(w) = \min\left\{1, \frac{(1 + (1-p)^3(1-(1-p)^{w-3}))}{(1-(1-p)^w)/(1-(1-p)^3)}\right\}. \qquad (5.17)$$

where w represents the window size. The probability that a sender detects a loss through a triple duplicate acknowledgement is therefore *1-Q(w)*. The expected delay corresponding to the occurrence of a RTO, denoted as $E[Z^{T_0}]$ is also derived in [5]:

$$E\left[Z^{T_0}\right]=\frac{f(p)T_0}{1-p},$$

where

$$f(p)=1+p+2p^2+4p^3+8p^4+16p^5+32p^6. \tag{5.18}$$

where T_0 denotes the average duration of the first time out in a sequence of one or more successive time outs. To estimate the value of T_0, we assume that the *RTT* is unaltered during the data transmission. Under this assumption, Equation (2.7) can be used for predicting the RTO after *N* times updating. It is easy to verify that maximum value of (2.7) is equal to $3RTT$. To reduce the complexity of computing T_0, we set in our simulation the minimum RTO to at least $4RTT$. Therefore T_0 can be estimated by the minimum RTO. From now we follow this assumption, i.e, $T_0 = \min RTO$.

In addition, the expected delay corresponding to the fast recovery is approximated by a single *RTT*, see [11]. Combining the above results, the expected delay $E[T_{loss}]$ due to any RTO or a fast recovery that happens at the end of the initial SS satisfies

$$E\left[T_{loss}\right]=l_{ss}\left(Q\left(E\left[W_{ss}\right]\right)E\left[Z^{T_0}\right]+\left(1-Q\left(E\left[W_{ss}\right]\right)\right)RTT\right). \tag{5.19}$$

5.1.4 *Transfer of the remaining data*

After the slow start phase and the fast retransmission/fast recovery after the first loss, TCP reaches the Congestion Avoidance phase where [11] suggests to use the goodput model from [10] to determine the time for transferring the remainder of the data.

The amount of data left after the initial SS and the recovery after the first loss is approximately

$$E\left[d_{ca}\right]=d-E\left[d_{ss}\right]. \tag{5.20}$$

The goodput, which is denoted by R, depends on the packet loss rate *p*, the average Retransmission Timer (T_0), the round Trip Time (*RTT*), the number of packets acknowledged per ACK and the maximum window (W_{max}). It is expressed as follows:

$$R=\begin{cases}\dfrac{\dfrac{1-p}{p}+\dfrac{W(p)}{2}+Q(W(p))}{RTT\left(\dfrac{b}{2}W(p)+1\right)+Q(W(p))E\left[Z^{T_0}\right]} & \text{if } W(p)<W_{\max}\\[2ex] \dfrac{\dfrac{1-p}{p}+\dfrac{W_{\max}}{2}+Q(W_{\max})}{RTT\left(\dfrac{b}{8}W_{\max}+\dfrac{1-p}{pW_{\max}}+2\right)+Q(W_{\max})E\left[Z^{T_0}\right]} & \text{otherwise}\end{cases}, \tag{5.21}$$

where *Q(w)* and $E[Z^{T_0}]$ are defined in (5.17) and (5.18), respectively, and where *W(p)* stands for the expected window size at the time of loss events when in congestion avoidance. It is expressed as follows:

$$W(p)=\frac{2+b}{3b}+\sqrt{\frac{8(1-p)}{3bp}+\left(\frac{2+b}{3b}\right)^2}. \tag{5.22}$$

Combine (5.20), (5.21) and (5.22), then $E[T_{ca}]$ as the expected time to send the remaining data is estimated as

$$E[T_{ca}] = \frac{E[d_{ca}]}{R(p, RTT, T_0, W_{\max})}. \tag{5.23}$$

Note that as already mentioned in the previous subsection we assume $T_0 = \min RTO$.

5.1.5 *Complete model for the mean total data download time*

The complete model for the mean Data Transfer Time, denoted by $E[T_{DTT}]$, suggested in [11] is obtained from adding the four contributions described in the previous subsections. Therefore we have

$$E[T_{DTT}] = E[T_{ss}] + E[T_{loss}] + E[T_{ca}] + E[T_{delack}], \tag{5.24}$$

where $E[T_{ss}], E[T_{loss}], E[T_{ca}]$, and $E[T_{delack}]$ are given in (5.14), (5.19), (5.23), and (5.15) respectively. For computing the mean TDT, we need to add up the TCP connection establishment time and the DTT. As mentioned at the beginning of this chapter, the connection establishment time can be approximated by ¾ times the RT. Therefore, the mean TDT can be expressed as follows:

$$TDT = E[T_{DTT}] + \frac{3}{4} L, \tag{5.25}$$

where DTT is given in (5.24) and L is given in (4.5).

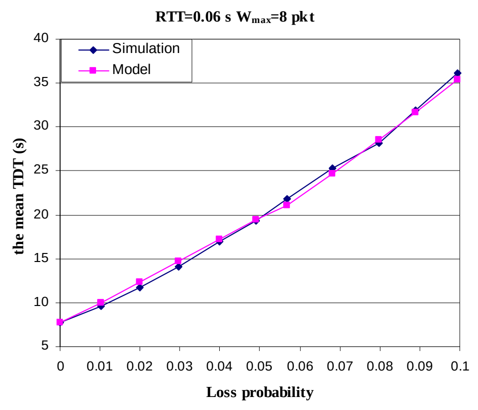
5.2 Numerical results

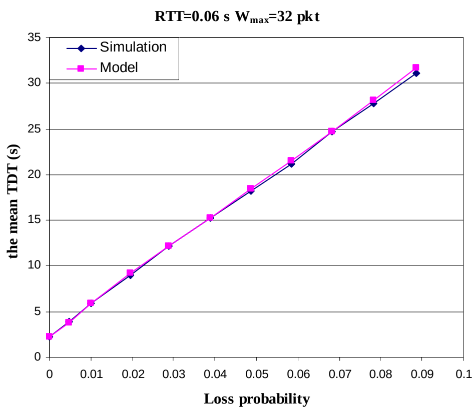
In this section we report a ns-2 simulation experiment that has been run as a validation for the Cardwell's model to predict the TDT.

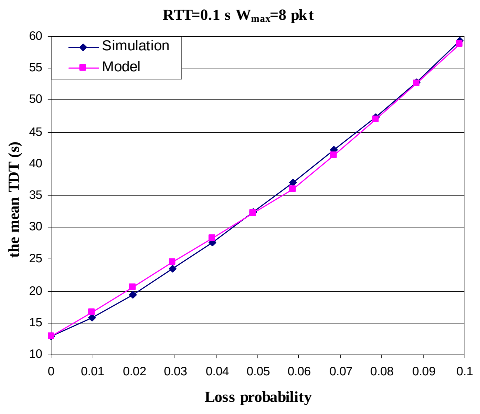
For the TCP parameters we choose the following values:

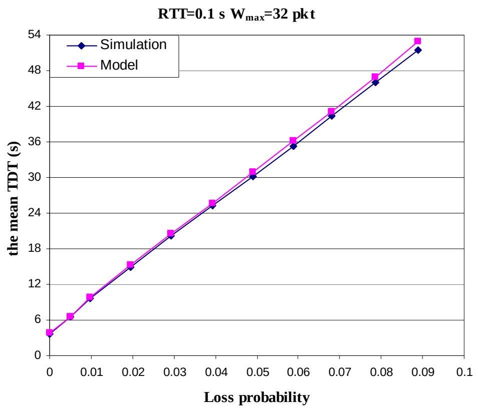
- Maximum segment size (*MSS*) : 1000 Bytes
- Maximum advertised window size (W_{max}): {8, 32} packets
- The number of data packets acknowledged by one ACK (*b*): 1
- The initial SS window size (w_1): 1 packet
- Access-link capacity: 30 Mbps
- Bottleneck-link capacity: 200 Mbps
- Fixed round-trip time (*RTT*): {0.06, 0.1, 0.3, 0.5} seconds
- Packet loss probability (*p*): {0, 0.01, …, 0.1}
- File size: 1000 packets
- Minimum RTO: $4RTT$

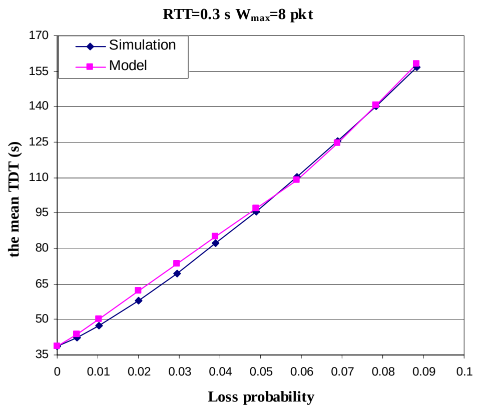
In [19], it is mentioned that $RTO = 4RTT$ works reasonably well to provide fairness with TCP. Therefore in the simulation, we set the minimum RTO to 4*RTT*. The simulation results are obtained by performing 10 independent simulation runs in which 100 servers are active to send the file with 1000 data segments. The simulation topology is the same as the topology illustrated in Figure 3-7. In each run, the mean TDT of 100 servers is obtained and finally the average TDT of the 10 individual runs is taken to compare the TDT determined by the Cardwell model discussed in Section 5.1. The comparison outcomes are shown in Figure 5-1 and also provided in Table 5-1. To ascertain the accuracy of the model, we compute the relative error of the model for the given packet loss probability and append the results into Table 5-1.

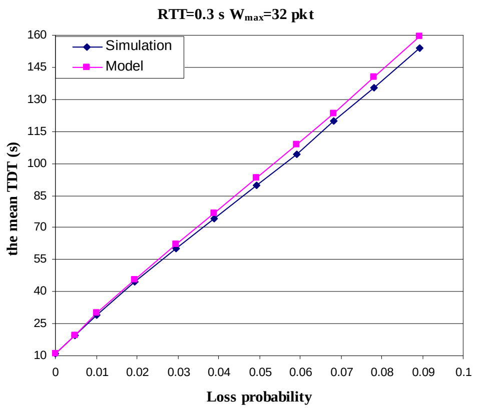


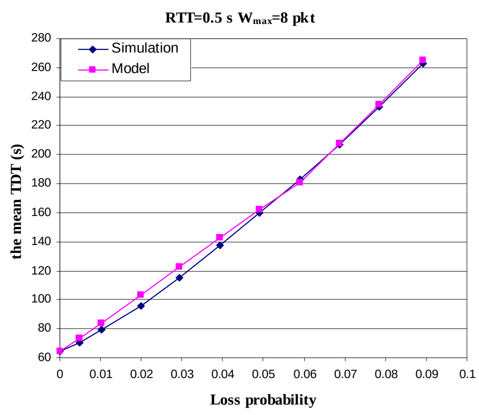












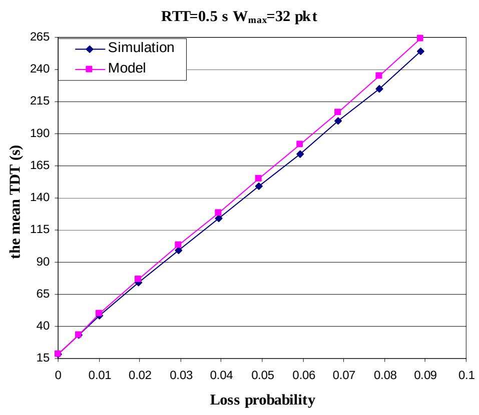


Figure 5-1: Comparing the mean TDT predicted by the Cardwell model to the mean TDT obtained by the simulation

	RTT = 0.06 s, Wmax = 8			RTT = 0.06 s, Wmax = 32		
p	NS	Model Cardwell		NS	Model Cardwell	
	the mean TDT	the mean TDT	ε_{rel}	the mean TDT	the mean TDT	ε_{rel}
0	7.78	7.77	**0%**	2.23	2.25	**1%**
0.01	9.57	10.03	**5%**	3.96	3.84	**-3%**
0.02	11.68	12.36	**6%**	5.87	5.89	**0%**
0.03	14.15	14.72	**4%**	9.04	9.17	**1%**
0.04	17.01	17.22	**1%**	12.19	12.20	**0%**
0.05	19.35	19.41	**0%**	15.21	15.31	**1%**
0.06	21.81	21.06	**-3%**	18.20	18.43	**1%**
0.07	25.34	24.64	**-3%**	21.21	21.58	**2%**
0.08	28.21	28.57	**1%**	24.70	24.71	**0%**
0.09	31.92	31.70	**-1%**	27.81	28.08	**1%**
0.1	36.19	35.44	**-2%**	31.14	31.65	**2%**
	RTT = 0.1 s, Wmax = 8			RTT = 0.1 s, Wmax = 32		
p	NS	Model Cardwell		NS	Model Cardwell	
	the mean TDT	the mean TDT	ε_{rel}	the mean TDT	the mean TDT	ε_{rel}
0	12.94	12.95	**0%**	3.71	3.75	**1%**
0.01	15.86	16.62	**5%**	6.59	6.56	**0%**
0.02	19.40	20.58	**6%**	9.61	9.77	**2%**
0.03	23.46	24.50	**4%**	14.93	15.25	**2%**
0.04	27.65	28.36	**3%**	20.19	20.52	**2%**
0.05	32.35	32.30	**0%**	25.21	25.66	**2%**
0.06	36.97	36.04	**-3%**	30.13	30.90	**3%**
0.07	42.14	41.28	**-2%**	35.24	36.10	**2%**
0.08	47.37	46.91	**-1%**	40.37	41.11	**2%**
0.09	52.80	52.71	**0%**	45.94	46.93	**2%**
0.1	59.36	58.73	**-1%**	51.54	52.87	**3%**
	RTT = 0.3 s, Wmax = 8			RTT = 0.3 s, Wmax = 32		
p	NS	Model Cardwell		NS	Model Cardwell	
	the mean TDT	the mean TDT	ε_{rel}	the mean TDT	the mean TDT	ε_{rel}
0	38.74	38.85	**0%**	11.11	11.25	**1%**
0.01	42.48	43.74	**3%**	19.73	19.43	**-1%**
0.02	47.43	50.13	**6%**	28.96	29.84	**3%**
0.03	57.81	61.98	**7%**	44.56	45.76	**3%**
0.04	69.40	73.58	**6%**	59.97	61.98	**3%**
0.05	82.41	85.09	**3%**	74.28	76.71	**3%**
0.06	95.46	97.10	**2%**	89.78	93.04	**4%**
0.07	110.51	108.71	**-2%**	104.49	109.03	**4%**
0.08	125.32	124.43	**-1%**	119.85	123.58	**3%**
0.09	140.14	140.67	**0%**	135.60	140.26	**3%**
0.1	156.47	157.93	**1%**	154.11	159.33	**3%**

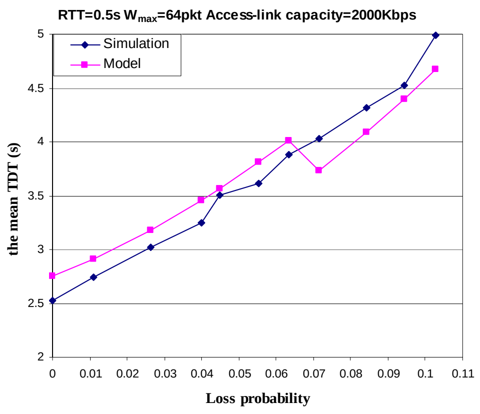
	RTT = 0.5 s, Wmax = 8			RTT = 0.5 s, Wmax = 32		
p	NS	Model Cardwell		NS	Model Cardwell	
	the mean TDT	the mean TDT	ε_{rel}	the mean TDT	the mean TDT	ε_{rel}
0	64.54	64.75	**0%**	18.51	18.75	**1%**
0.01	70.80	73.19	**3%**	33.07	33.03	**0%**
0.02	79.16	83.98	**6%**	48.11	49.62	**3%**
0.03	95.82	103.33	**8%**	74.29	76.70	**3%**
0.04	115.39	122.29	**6%**	99.55	103.03	**4%**
0.05	137.32	142.70	**4%**	124.06	128.56	**4%**
0.06	159.64	161.98	**1%**	149.20	154.87	**4%**
0.07	183.06	181.06	**-1%**	174.00	181.81	**4%**
0.08	207.27	207.50	**0%**	199.91	206.84	**3%**
0.09	233.05	234.57	**1%**	224.63	235.03	**5%**
0.1	262.91	265.29	**1%**	254.16	263.83	**4%**

Table 5-1: A comparison of the mean TDT depicted by the Cardwell model and the mean TDT obtained from the simulation

Figure 5-1 and Table 5-1 clearly show that the Cardwell model for predicting TDT is quite accurate in the case of large files. Be aware that in the case of $p = 0$, the mean TDT can be computed by (5.14). Another note is that by observing the figure, it seems that the mean TDT growths linearly. But that is only for low *p*. When *p* is a larger value, the mean TDT will growth exponentially. In order to confirm the accuracy of this model for small files, we simulate more scenarios and compare those results to the outcomes obtained from the model. For new scenarios, we choose the following values as TCP parameters:

- Maximum segment size (*MSS*) : 1000 Bytes
- Maximum advertised window size (W_{max}): 64 packets
- The number of data packets acknowledged by one ACK (*b*): 1
- The initial SS window size (w_1): 1 packet
- Access-link capacity: {2000, 512} Kbps
- Bottleneck-link capacity: 200 Mbps
- Fixed round-trip time (*RTT*): 0.5 seconds
- Packet loss probability (*p*): {0, 0.01, …, 0.1}
- File size: 10 packets
- Minimum RTO: 1 second

The related outcomes obtained from the model and the simulation are shown in Figure 5-2 and Table 5-2.



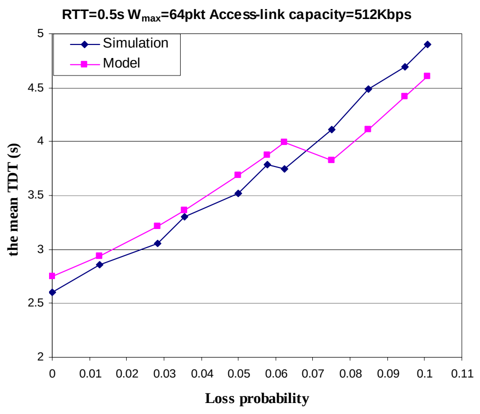


Figure 5-2: Comparison results for new scenarios

	RTT = 0.5 s, Wmax = 64, Access-link capacity=2000kbps			RTT = 0.5 s, Wmax = 64, Access-link capacity=512kbps		
p	NS	Model Cardwell		NS	Model Cardwell	
	the mean TDT	the mean TDT	ε_{rel}	the mean TDT	the mean TDT	ε_{rel}
0	2.53	2.75	**9%**	2.60	2.75	**6%**
0.01	2.74	2.91	**6%**	2.86	2.93	**3%**
0.02	3.02	3.18	**5%**	3.05	3.22	**5%**
0.03	3.24	3.46	**7%**	3.30	3.36	**2%**
0.04	3.51	3.57	**2%**	3.52	3.69	**5%**
0.05	3.61	3.81	**6%**	3.78	3.87	**2%**
0.06	3.88	4.01	**3%**	3.75	3.99	**7%**
0.07	4.03	3.73	**-7%**	4.11	3.82	**-7%**
0.08	4.32	4.09	**-5%**	4.49	4.11	**-9%**
0.09	4.52	4.40	**-3%**	4.70	4.41	**-6%**
0.1	4.99	4.67	**-6%**	4.91	4.61	**-6%**

Table 5-2: Numerical comparison results for new scenarios

From Figure 5-2 and Table 5-2, we notice that in the new scenarios the mean TDT obtained by the model is close to the mean TDT from the simulation. For $p \le 10\%$ the relative errors of the model under these new conditions are all within ±10%. In conclusion the Cardwell model for predicting the TDT is also quite accurate for small files.

5.3 Extended model

The Cardwell model discussed in Section 5.1 assumes that only data packets sent from the sender to the receiver can be lost during the data transmission. This assumption ignores the fact that the ACKs for received data packets can also be dropped. In this section we extend the model by including the impact of the loss of the ACKs.

First, we define the new variables introduced in this section:

p_f :	forward packet loss rate. This is the data packet loss rate from the sender to the receiver
p_b :	backward packet loss rate. This is the ACK loss rate from the receiver to the sender
w :	the current window size

As a first step, we discuss in which situation a data packet is considered lost by TCP. We focus on one data packet. We assign to this data packet an index *k*. *k* indicates also the position of the data packet within the current window. Therefore *k* satisfies the condition $1 \le k \le w$. Obviously, if *k* is really lost during transmission from the sender to the receiver, then this packet is undoubtedly considered lost by the sender TCP. See Figure 5-3. Denote this case as Case 1. Then it is easy to verify that p_1, denoted as the probability of the occurrence of Case 1 is equal to p_f.

$$p_1 = p_f \tag{5.26}$$

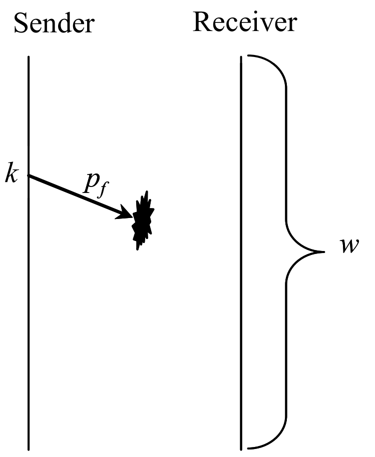


Figure 5-3: Case 1: packet *k* is lost from the sender to the receiver

Consider another situation wherein packet *k* is sent successfully, but the ACK for packet *k* containing the information of expecting packet *k+1,* is lost, see Figure 5-4. Denote this case as Case 2. Then p_2, the probability of the occurrence of Case 2, can be computed as,

$$p_2 = (1 - p_f)p_b \tag{5.27}$$

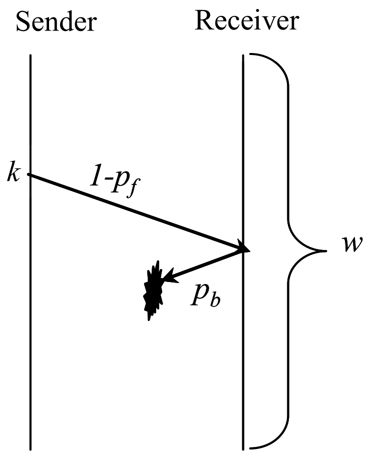


Figure 5-4: Case 2: packet *k* is sent successfully from the sender to the receiver, but its ACK is lost in the reverse direction

For Case 2, we cannot determine immediately whether data packet *k* is considered lost by the sender TCP or not. To find that answer, we need to analyse the TCP behaviour within a window size. In [11], it is assumed that the time needed to send all the data packets within a window is much smaller than the duration of a round. This implies that packet *k+1* is sent before receiving the ACK of data packet *k*. Therefore the loss of the ACK for packet *k* has no impact on triggering sending packet *k+1*. But the ACK for packet *k+1* has influence on determining if packet *k* is sent successfully.
First, we take a close look at the situation that packet *k+1* is sent successfully and its ACK is received by the sender. According to Case 2, packet *k* is already received by the receiver. Therefore the ACK for packet *k+1* sent by the receiver contains the information that packet *k+2* is expected instead that packet *k* is expected. Therefore, this ACK reports the TCP sender that packet *k+1* is sent and received successfully. Indirectly, this ACK confirms also that packet *k* is arrived at the receiver. That is because of the fact that the TCP sender considers that packet *k* is lost only if it receives the ACKs containing the information that packet *k* is expected. But these ACKs cannot be generated by the receiver under the assumption of Case 2.
Then, we discuss another situation wherein the ACK for packet *k+1* is lost during the transmission. This situation can be caused by loss of packet *k+1* or by loss of the ACK for packet *k+1*. Denote the probability for this situation as p_3. Then we have,

$$p_3 = p_f + (1 - p_f)p_b \tag{5.28}$$

In this situation, we need to investigate further if the ACK for packet *k+2* is received by the sender. The information that the ACK for packet *k+2* contains is not certain. It depends on whether packet *k+1* is received by the receiver. But anyway, under the condition of Case 2, the expected packet number contained in this ACK is at least *k+1*. Therefore if the ACK for packet *k+2* is received by the sender, the sender is ensured that the packet *k* is received by the receiver. If this ACK is not received by the sender, then we have to investigate the ACK for packet *k+3*. Then we repeat the analysis mentioned above.
In conclusion, under Case 2, if $\exists d \in [1, w-k]$, the ACK of packet *k+d* is received by the sender, then the sender is ensured that packet *k* is sent successfully. For this situation, Figure 5-5 illustrates an example. If for $\forall d \in [1, w-k]$, the ACK of packet *k+d* is not received by the sender, packet *k* is considered lost by the sender TCP, see Figure 5-6.

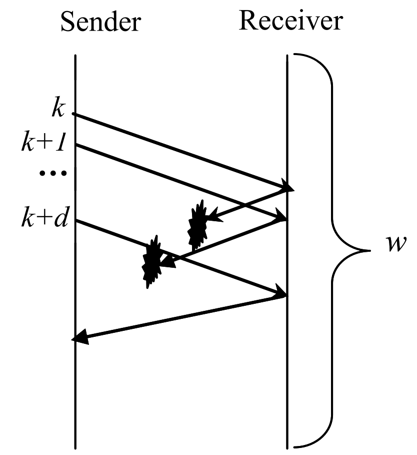


Figure 5-5: Packet *k* is considered sent successfully by the sender

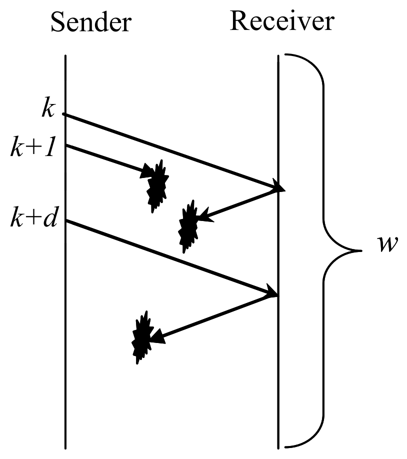


Figure 5-6: Packet *k* is considered lost by the sender

Denote p_k as the probability that packet *k* is considered lost by the sender. Based on (5.28), we have,

$$p_k = \left[p_f + (1-p_f)p_b\right]^{w-k}. \tag{5.29}$$

Therefore, under Case 2, the expected probability that a packet is considered lost by the sender, denoted by $E\left[p_{case2}\right]$, can be expressed as follows:

$$E\left[p_{case2}\right] = \sum_{k=1}^{w} \frac{p_k}{w} = \sum_{k=1}^{w} \frac{\left[p_f + (1-p_f)p_b\right]^{w-k}}{w}. \tag{5.30}$$

Combine Case 1 and Case 2, the complete expression for *p*, denoting the probability that a packet is considered lost by the sender, can be deduced. Based on (5.26), (5.27), and (5.30), we have

$$p = p_f + (1-p_f)p_b \sum_{k=1}^{w} \frac{\left[p_f + (1-p_f)p_b\right]^{w-k}}{w}. \tag{5.31}$$

(5.31) can be simplified as follows:

$$p = p_f + \frac{p_b}{w(1-p_b)}\left[1 - (p_f + p_b - p_f p_b)^w\right]. \tag{5.32}$$

The expression of (5.32) includes the unknown variable *w*. At the beginning of this section, we defined *w* as the current window size. But in fact, we should not just focus on one window size or on one TCP round. If the ACKs for the packets sent before packet *k* are received by the sender, then that triggers sending new data packets in the new round. The ACKs of the newly sent data packets have also influence on determining whether packet *k* is considered lost by the sender or not. Therefore, we redefine *w* as the expected number of packets sent up to and including the first lost packet. Because p_3 defined in (5.28) is also the packet loss probability under the assumption that packet loss is independent, we approximate *w* as:

$$w = \sum_{k=1}^{l} k \times (1-p_3)^{k-1} \times p_3 + (1-p_3)^l \times l = \frac{1-(1-p_3)^l}{p_3}, \tag{5.33}$$

where p_3 is given in (5.28) and l represents the number of remaining data packets to send when a packet loss occurs. Note that l reduces during the data transmission. Since the value of l is between 0 and d, where d stands for the file size, for simplification, we approximate l as $d/2$,

We adapt the Cardwell model described in Section 5.1 by substituting the packet loss probability given in (5.32). Then we obtain the extended model for predicting the mean TDT for the case that the packet loss happens not only on the link from the sender to the receiver but also on the link in the reverse direction.

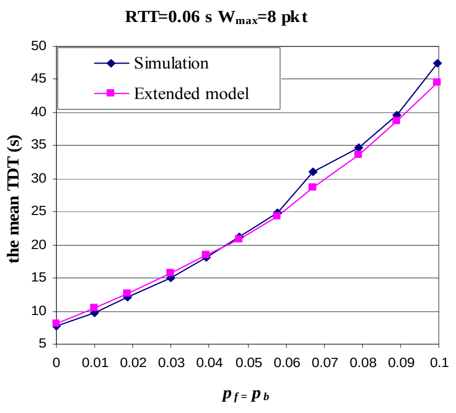
5.4 Numerical results

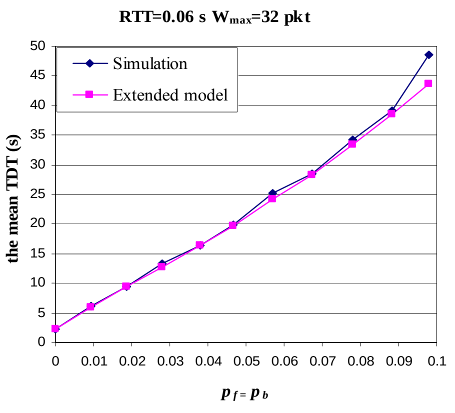
In this section we report ns-2 simulation experiments that have been run as a validation for the extended model. In the simulation, the forward loss probability is set equal to the backward loss probability.

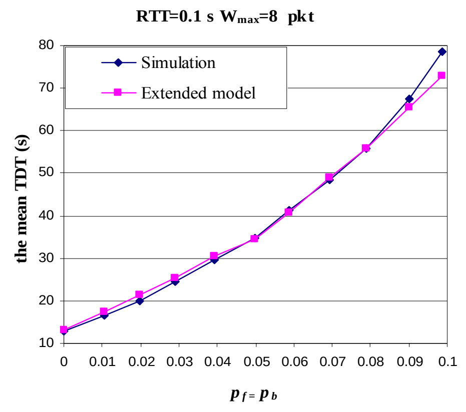
For the TCP parameters we choose the following values:

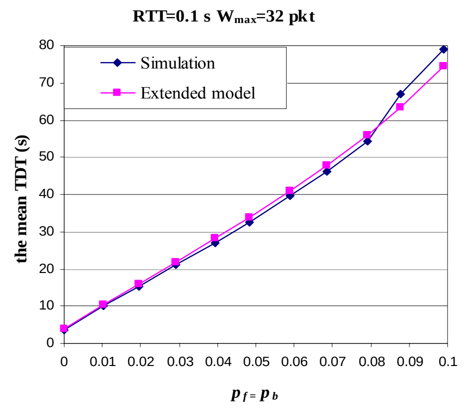
- Maximum segment size (*MSS*) : 1640 Bytes
- Maximum advertised window size (W_{max}): {8,32} packets
- The number of data packets acknowledged by one ACK (b): 1
- The initial SS window size (w_1): 1 packet
- Access-link capacity: 30 Mbps
- Bottleneck-link capacity: 200 Mbps
- Fixed round-trip time (*RTT*): {0.06, 0.1, 0.3, 0.5} seconds
- Forward loss probability (p_f): {0, 0.01, …, 0.1}
- Backward loss probability (p_b): p_f
- File size: 1000 packets
- Minimum RTO: $4RTT$

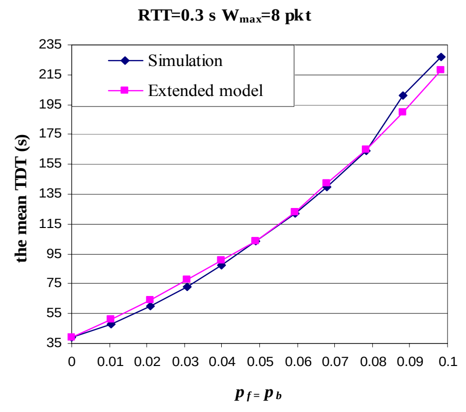
The results of these experiments are provided in Figure 5-7 and Table 5-3.

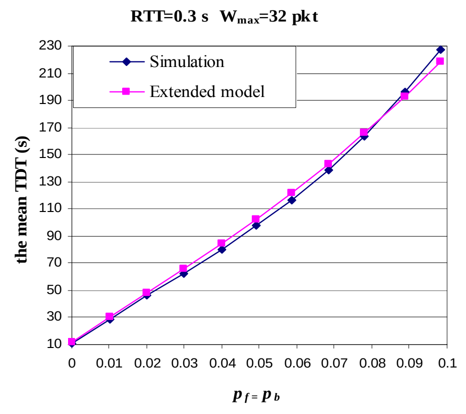


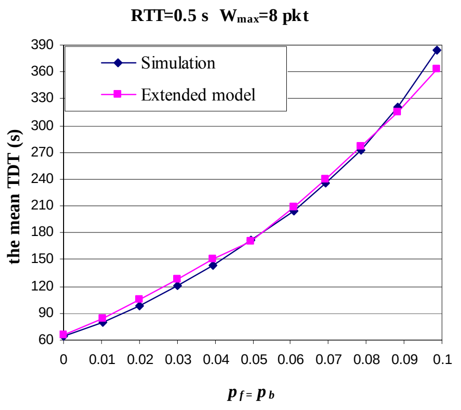



RTT=0.1 s W_{max}=8 pkt
Simulation
Extended model
the mean TDT (s)
$p_f = p_b$
80
70
60
50
40
30
20
10
0
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
0.09
0.1


RTT=0.1 s W_{max}=32 pkt
Simulation
Extended model
the mean TDT (s)
$p_f = p_b$
80
70
60
50
40
30
20
10
0
0
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
0.09
0.1


RTT=0.3 s W_{max}=8 pkt
Simulation
Extended model
the mean TDT (s)
$p_f = p_b$
235
215
195
175
155
135
115
95
75
55
35
0
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
0.09
0.1


RTT=0.3 s W_{max}=32 pkt
Simulation
Extended model
the mean TDT (s)
$p_f = p_b$
230
210
190
170
150
130
110
90
70
50
30
10
0
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
0.09
0.1



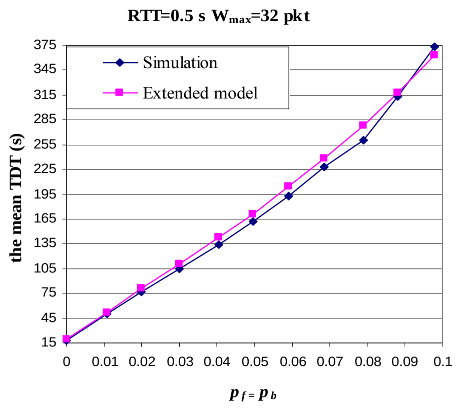


Figure 5-7: Comparing the extended model for predicting the mean TDT to simulation results in case of two-way loss

	RTT = 0.06 s, Wmax = 8			RTT = 0.06 s, Wmax = 32		
p_f=p_b	NS	Extended Cardwell Model		NS	Extended Cardwell Model	
	the mean TDT	the mean TDT	ε_{rel}	the mean TDT	the mean TDT	ε_{rel}
0	7.81	8.15	**4%**	2.24	2.30	**3%**
0.01	9.79	10.38	**6%**	6.10	5.92	**-3%**
0.02	12.15	12.72	**5%**	9.34	9.45	**1%**
0.03	14.97	15.68	**5%**	13.34	12.69	**-5%**
0.04	18.08	18.55	**3%**	16.35	16.38	**0%**
0.05	21.19	20.85	**-2%**	19.97	19.71	**-1%**
0.06	24.78	24.30	**-2%**	25.14	24.10	**-4%**
0.07	31.09	28.63	**-8%**	28.49	28.35	**-1%**
0.08	34.78	33.57	**-3%**	34.25	33.39	**-3%**
0.09	39.57	38.64	**-2%**	39.17	38.58	**-2%**
0.1	47.47	44.49	**-6%**	48.53	43.61	**-10%**
	RTT = 0.1 s, Wmax = 8			RTT = 0.1 s, Wmax = 32		
p_f=p_b	NS	Extended Cardwell Model		NS	Extended Cardwell Model	
	the mean TDT	the mean TDT	ε_{rel}	the mean TDT	the mean TDT	ε_{rel}
0	12.97	13.26	**2%**	3.72	3.83	**3%**
0.01	16.44	17.39	**6%**	10.08	10.37	**3%**
0.02	20.01	21.50	**7%**	15.37	15.96	**4%**
0.03	24.43	25.49	**4%**	21.22	21.77	**3%**
0.04	29.54	30.44	**3%**	27.15	28.21	**4%**
0.05	34.82	34.57	**-1%**	32.41	33.71	**4%**
0.06	41.37	40.67	**-2%**	39.70	40.82	**3%**
0.07	48.49	49.08	**1%**	46.28	47.73	**3%**
0.08	55.84	55.79	**0%**	54.47	55.96	**3%**
0.09	67.52	65.35	**-3%**	67.03	63.38	**-5%**
0.1	78.59	72.75	**-7%**	78.87	74.50	**-6%**

	RTT = 0.3 s, Wmax = 8			RTT = 0.3 s, Wmax = 32		
p_f=p_b	NS	Extended Cardwell Model		NS	Extended Cardwell Model	
	the mean TDT	the mean TDT	ε_{rel}	the mean TDT	the mean TDT	ε_{rel}
0	38.77	39.09	**1%**	11.12	11.38	**2%**
0.01	48.01	50.97	**6%**	28.93	30.44	**5%**
0.02	59.79	64.19	**7%**	46.22	48.48	**5%**
0.03	73.10	77.54	**6%**	62.46	65.76	**5%**
0.04	87.37	90.63	**4%**	80.43	84.26	**5%**
0.05	103.26	103.38	**0%**	98.10	102.47	**4%**
0.06	121.97	123.20	**1%**	116.14	121.66	**5%**
0.07	139.58	141.97	**2%**	138.67	142.78	**3%**
0.08	164.05	165.22	**1%**	163.86	165.98	**1%**
0.09	200.73	189.52	**-6%**	196.40	193.08	**-2%**
0.1	226.64	217.70	**-4%**	226.97	218.90	**-4%**
	RTT = 0.5 s, Wmax = 8			RTT = 0.5 s, Wmax = 32		
p_f=p_b	NS	Extended Cardwell Model		NS	Extended Cardwell Model	
	the mean TDT	the mean TDT	ε_{rel}	the mean TDT	the mean TDT	ε_{rel}
0	64.57	65.08	**1%**	18.52	18.85	**2%**
0.01	79.85	84.60	**6%**	49.83	52.02	**4%**
0.02	98.02	105.26	**7%**	76.60	80.58	**5%**
0.03	121.31	127.90	**5%**	105.14	110.24	**5%**
0.04	143.83	150.08	**4%**	133.53	142.24	**7%**
0.05	171.70	170.54	**-1%**	161.86	171.44	**6%**
0.06	204.82	208.31	**2%**	193.39	204.38	**6%**
0.07	235.36	239.19	**2%**	228.33	238.56	**4%**
0.08	272.42	276.02	**1%**	260.41	277.71	**7%**
0.09	320.76	315.16	**-2%**	312.72	318.02	**2%**
0.1	384.61	363.39	**-6%**	372.91	362.54	**-3%**

Table 5-3: Comparing the extended model for predicting the mean TDT to simulation results in case of two-way loss

From Figure 5-7 and Table 5-3, we conclude that the extended Cardwell model is very accurate for predicting the mean TDT in the case that packet loss occurs both in the direction from the sender to receiver and vice versa. In order to confirm the accuracy of this model, we simulate more scenarios and compare those results to the outcomes obtained from the model. For the new scenarios, we choose the following values as TCP parameters:

- Maximum segment size (*MSS*) : 1640 Bytes
- Maximum advertised window size (W_{max}): 64 packets
- The number of data packets acknowledged by one ACK (b): 1
- The initial SS window size (w_1): 1 packet
- Access-link capacity: {2000, 512} Kbps
- Bottleneck-link capacity: 200 Mbps
- Fixed round-trip time (*RTT*): 0.5 seconds
- Forward loss probability (p_f): {0, 0.01, …, 0.1}
- Backward loss probability (p_b): p_f
- File size: {10, 1000} packets
- Minimum RTO: $4RTT$

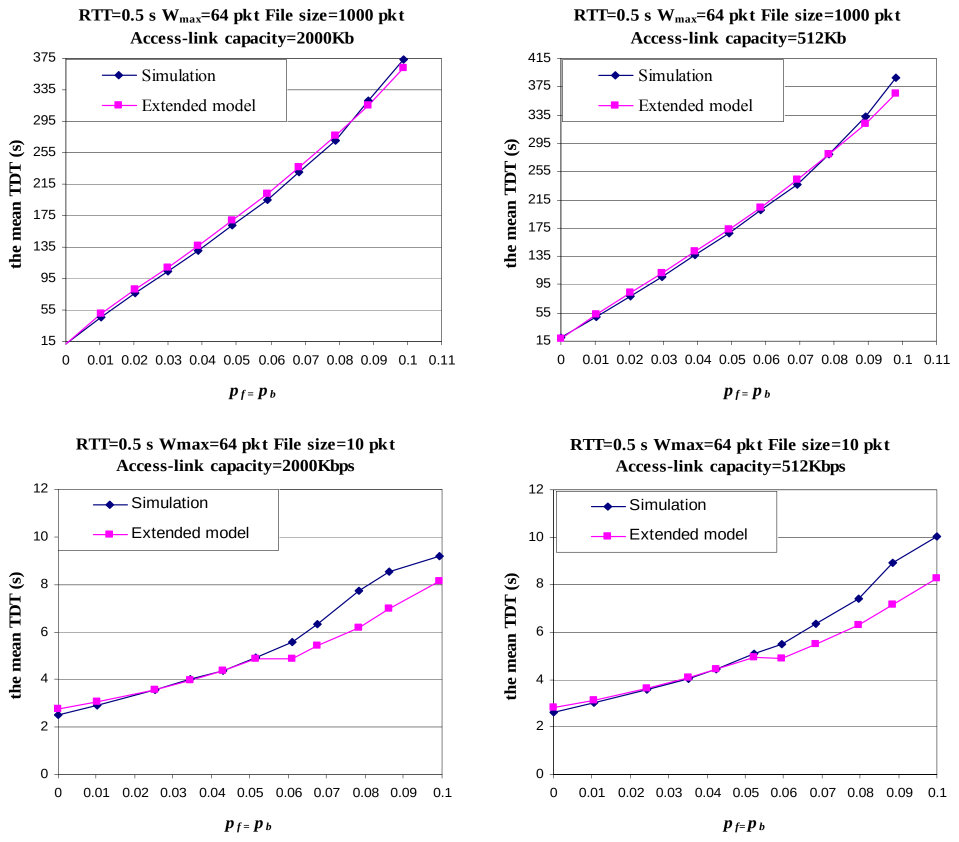


Figure: 5-8: Comparison of the mean TDT under the new chosen maximum window size and the access-link capacity

File size=1000	RTT = 0.5 s, Wmax = 64, Access-link capacity=2000kbps			RTT = 0.5 s, Wmax = 64, Access-link capacity=512kbps		
p_f=p_b	NS	Extended Cardwell Model		NS	Extended Cardwell Model	
	the mean TDT	the mean TDT	ε_{rel}	the mean TDT	the mean TDT	ε_{rel}
0	11.26	11.36	**1%**	19.33	18.46	**-5%**
0.01	46.30	50.75	**10%**	48.98	52.73	**8%**
0.02	76.40	81.05	**6%**	78.83	84.07	**7%**
0.03	104.41	109.15	**5%**	106.69	110.84	**4%**
0.04	130.49	136.83	**5%**	136.29	141.24	**4%**
0.05	162.72	168.42	**4%**	168.11	172.54	**3%**
0.06	195.56	202.48	**4%**	200.05	204.37	**2%**
0.07	230.44	236.04	**2%**	235.77	242.89	**3%**
0.08	270.41	276.04	**2%**	278.75	279.22	**0%**
0.09	321.75	315.25	**-2%**	333.42	322.57	**-3%**
0.1	373.60	362.09	**-3%**	386.84	364.54	**-6%**
File size=10	RTT = 0.5 s, Wmax = 64, Access-link capacity=2000kbps			RTT = 0.5 s, Wmax = 64, Access-link capacity=512kbps		
p_f=p_b	NS	Extended Cardwell Model		NS	Extended Cardwell Model	
	the mean TDT	the mean TDT	ε_{rel}	the mean TDT	the mean TDT	ε_{rel}
0	2.53	2.78	**10%**	2.60	2.83	**9%**
0.01	2.92	3.07	**5%**	3.01	3.14	**4%**
0.02	3.55	3.57	**1%**	3.60	3.62	**1%**
0.03	4.01	3.98	**-1%**	4.02	4.08	**1%**
0.04	4.39	4.39	**0%**	4.46	4.43	**-1%**
0.05	4.92	4.87	**-1%**	5.09	4.96	**-3%**
0.06	5.57	4.87	**-12%**	5.48	4.91	**-10%**
0.07	6.33	5.43	**-14%**	6.37	5.52	**-13%**
0.08	7.75	6.19	**-20%**	7.39	6.32	**-15%**
0.09	8.53	7.00	**-18%**	8.90	7.15	**-20%**
0.1	9.16	8.14	**-11%**	10.02	8.25	**-18%**

Table 5-4: Comparison of the mean TDT under the new chosen maximum window size and the access-link capacity

Figure 5-8 and Table 5-4 show that the extended model works very well for predicting the mean TDT for the file with 1000 data packets in the new scenarios. The relative

errors under these new situations are very low; in most cases they are within ±6%. For the file with only 10 data packets, the extended model is very accurate under the condition that the forward and backward loss probabilities are both lower than or equal to 5%. But for both probabilities higher than 5%, the model seems not very accurate. Note that the probability that a packet is considered lost is higher than the one way loss probability. For $p_f = p_b = 0.07$, from (5.32), we obtain $p = 0.09$. For this case, the relative errors shown in the table are not very high (namely -14% and -13% respectively). Therefore, we can conclude that the model performs quite well for a small file.
The experiments mentioned above are all under the assumption that the forward loss probability is equal to the backward loss probability. But in a realistic network environment, the link used for sending the data packets is not always the same as the link used for sending of the ACKs. This implies that the packet loss probabilities belonging to these two links are not necessarily equivalent. In addition, ACKs are much smaller than data packets. Therefore the buffer for ACKs fills less rapidly than the buffer for data packets when these two buffers have the same size. This might result in different packet drop probabilities. To confirm if the extended model is also suitable for these situations, we simulate another scenario in which two different loss probabilities are simulated.

The parameter settings for this experiment are:

- Maximum segment size (*MSS*) : 1000 Bytes
- Maximum advertised window size (W_{max}): 32 packets
- The number of data packets acknowledged by one ACK (b): 1
- The initial SS window size (w_1): 1 packet
- Access-link capacity: 2000 Kbps
- Bottleneck-link capacity: 100 Mbps
- Fixed round-trip time (*RTT*): 0.3 seconds
- Forward loss probability (p_f): {0, 0.01, …, 0.1}
- Backward loss probability (p_b): 0.02
- File size: 500 packets
- Minimum RTO: $4RTT$

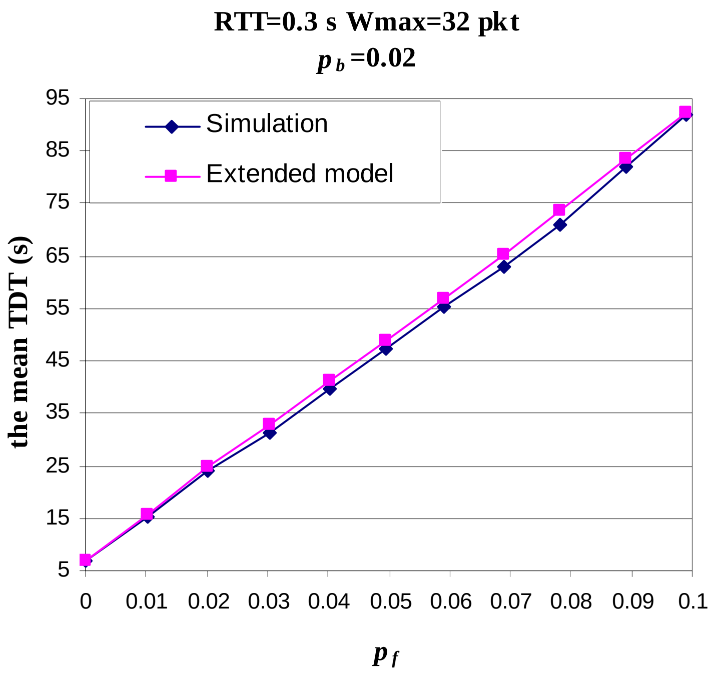


Figure 5-9: Comparison result for different packet loss probabilities of the forward link and backward link

	RTT = 0.3 s, Wmax = 32, p_b=0.02		
p_f	NS	Extended Cardwell Model	
	the mean TDT	the mean TDT	ε_{rel}
0	6.91	6.83	**-1%**
0.01	15.27	15.68	**3%**
0.02	23.95	24.65	**3%**
0.03	31.42	33.01	**5%**
0.04	39.60	41.31	**4%**
0.05	47.23	49.04	**4%**
0.06	55.24	57.04	**3%**
0.07	62.92	65.20	**4%**
0.08	70.82	73.50	**4%**
0.09	81.94	83.53	**2%**
0.1	91.96	92.38	**0%**

Table 5-5: Comparison result for different packet loss probabilities of the forward link and backward link

From Figure 5-9 and Table 5-5, we conclude that the extended model is also accurate for predicting the mean TDT under the situation that the forward packet loss probability is unequal to the backward packet loss probability.

6 Total Download Time in a multi-domain environment

In Chapter 5, the mean TDT model in a single-domain environment is discussed. In this chapter, we extend that model in a multi-domain setting and describe it in Section 6.1. In the subsequent section, the TDT model in a multi-domain environment is validated by means of ns-2 simulations.

6.1 The proposed model

In Chapter 5, we inferred that the mean TDT for downloading a file could be modelled as the function of the parameters d, $W_{\max}$, T_0, b, MSS, w_1, T_{delack}, RTT and p:

$$TDT = f_2\left(d; W_{\max}, T_0, b, MSS, w_1, T_{delack}; RTT, p\right), \tag{6.1}$$

where the function $f_2(.)$ is given explicitly in (5.25). All these parameters are defined in Chapter 5. For the readers' convenience, we list those definitions below,

d	file size
T_0	the average duration of the first time-out in a sequence of one or more successive time-outs at the server side
W_{max}	the maximum advertised window size
b	the number of data packets acknowledged by one ACK
MSS	the maximum data segment size
w_1	the initial window size
T_{delack}	the expected delay between the reception of a single segment and the delayed ACK for that segment
RTT	the round trip time
p	the probability that a packet is considered lost by the sender TCP

In (6.1), the parameter d is fixed and so are the TCP parameters b, MSS, w_1 and T_{delack}. The parameter W_{max} is set according to the sender's and the receiver's buffer size and the network bandwidth. The parameters RTT and p rely completely on the network condition. Finally T_0 is estimated as the minimum RTO.
Let us assume that there exist N network domains between the client and the server and in each network domain i, where $1 \le i \le N$, the following performance measures are specified.

p_{f_i}:	forward packet loss probability
p_{b_i}:	backward packet loss probability
RTT_i:	round trip time
B_i:	bandwidth limitation

Figure 6-1 shows an example.

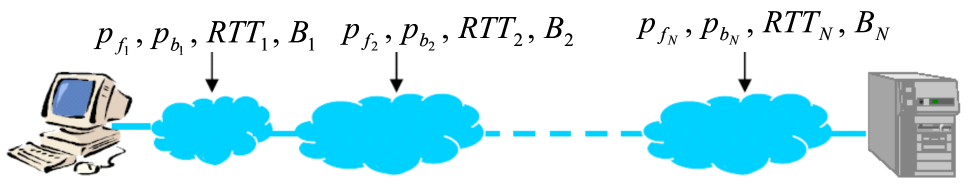

$p_{f_1}, p_{b_1}, RTT_1, B_1$
$p_{f_2}, p_{b_2}, RTT_2, B_2$
$p_{f_N}, p_{b_N}, RTT_N, B_N$

Figure 6-1: Packet loss probability p_i, round trip time RTT_i and possible bandwidth limitation (B_i) specifications per network domain

Under this assumption, we apply (6.1) in a multi-domain environment. Before that, the round trip time and packet loss probability in the entire network, denoted by *RTT* and *p* respectively, have to be determined. Denote p_f and p_b as the forward and the backward packet loss probability for the entire network. Then we have the following equations:

$$RTT = \sum_{i=1}^{N} RTT_i \ , \tag{6.2}$$

$$p_f = 1 - \prod_{i=1}^{N} (1 - p_{f_i}) \ , \tag{6.3}$$

$$p_b = 1 - \prod_{i=1}^{N} (1 - p_{b_i}) \ . \tag{6.4}$$

According to (5.32) and (5.33), *p* can be expressed as

$$p = p_f + \frac{p_b}{w(1 - p_b)} \left[1 - \left(p_f + p_b - p_f p_b \right)^w \right], \tag{6.5}$$

where *w* is given in (5.33).

Next, the impact of the bandwidth restriction in each domain on W_{max} must be quantified. The parameters W_{max} is derived from the bandwidth limitations B_i of the individual networks and the sender and the receiver buffer limitations. Denoting the latter by W_{send} and $W_{receiver}$, we approximate W_{max} as follows:

$$W_{\max} = \min \left\{ W_{send}, W_{recevier}, \frac{RTT}{MSS} B_1, \frac{RTT}{MSS} B_2, \ldots, \frac{RTT}{MSS} B_n \right\} . \tag{6.6}$$

Combining (6.1)-(6.6) we obtain the desired model for mean TDT for TCP based downloads in a multi-domain environment.

6.2 Numerical results

In this section the proposed model described in Section 6.1 is validated. The validation of the model is split into two steps. In the first step the model is validated under the assumption that only packets are dropped during the data transmission in the direction from the sender to the receiver. In the second step we validate the model in the case where a packet loss happens not only on the link from the sender to the receiver but also in the reverse direction. These two steps of the validation are described in Section 6.2.1 and Section 6.2.2 respectively. For both cases, we use the same topology shown in Figure 6-2. In the topology we assume that a user crosses three domains in order to reach a server.

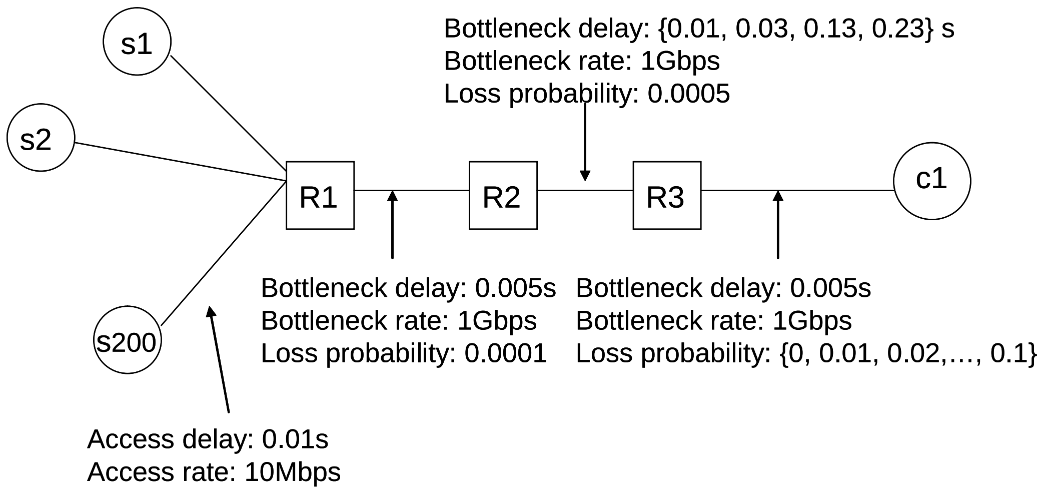


Figure 6-2: Simulation topology for the mean TDT model in a multi-domain environment

6.2.1 *Model validation when a packet loss only occurs from the sender to the receiver*

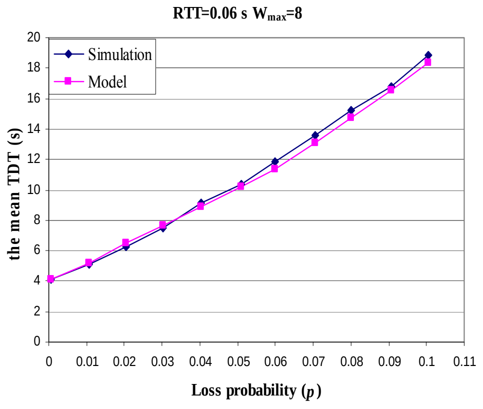
In this section, we assume that a packet loss occurs only from the sender to the receiver. This implies that all backward loss probabilities in each network domain are considered as 0. As shown in Figure 6-2, we have the following parameter settings for the simulations.

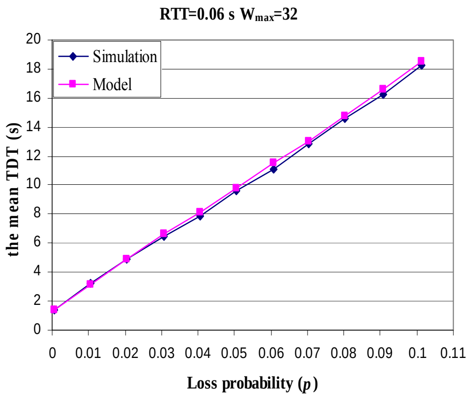
- *MSS*: 1640 Bytes
- W_{max}: {8, 32} packets
- *b*, the number of packets that is acknowledged by each ACK: 1
- w_1, the initial window size: 1
- Minimum RTO: 4 times the *RTT* of the entire network
- File size: 500 packets

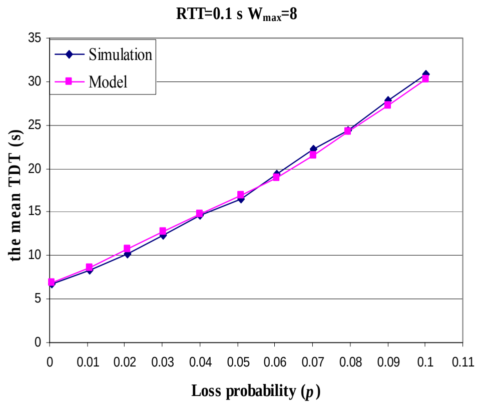
In the four network domains between the server and the client, we have the following assumptions. Note that for each network the round trip time is two times the bottleneck delay.

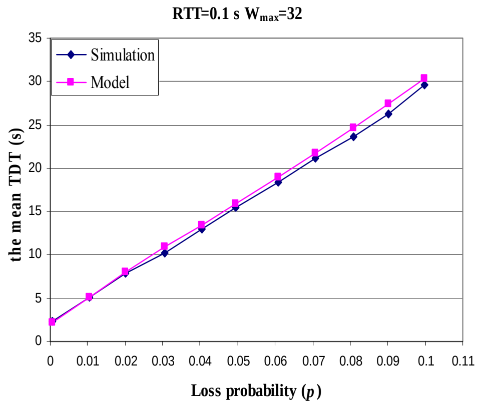
Network 1:
$p_{f_1}: 0,\ p_{b_1}: 0,\ RTT_1: 0.02s,\ B_1: 10Mbps$
Network 2
$p_{f_2}: 0.0001,\ p_{b_2}: 0,\ RTT_2: 0.01s\ B_2: 1Gbps$
Network 3
$p_{f_3}: 0.0005,\ p_{b_3}: 0,\ RTT_3: \{0.02, 0.06, 0.26, 0.46\}s,\ B_3: 1Gbps$
Network 4
$p_{f_4}: \{0, 0.01, ..., 0.1\},\ p_{b_4}: 0,\ RTT_4:\ 0.01s,\ B_4: 1Gbps$

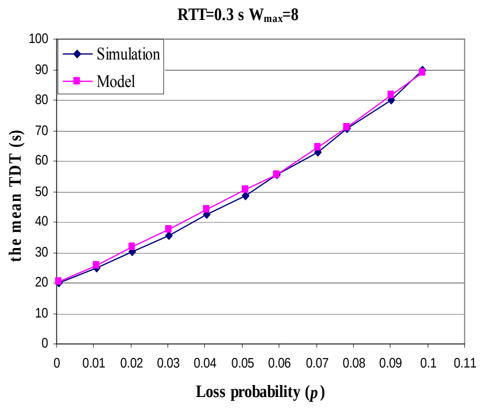
Network 1 typically corresponds to a broadband access network with more bandwidth than ADSL, e.g., VDSL. Networks 2, 3, and 4 typically correspond to core networks. For these parameter settings, the mean TDT can be estimated by the model discussed in the previous section. The estimated values are compared to the outcomes gained from the simulation. The comparison results are shown in Figure 6-3 and in Table 6-1. In order to make the difference between the estimated and simulated outcomes clearly, we compute the relative errors of the model under the different parameter settings and append them into Table 6-1. The mean TDT is given in the unit of seconds.

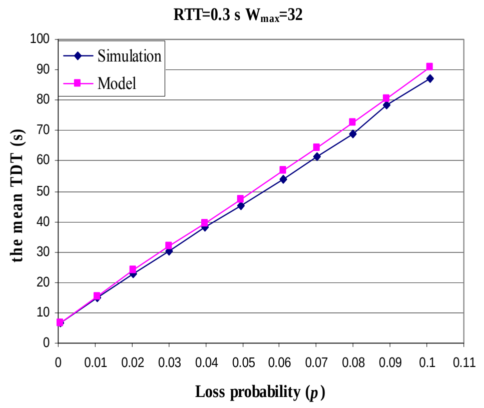


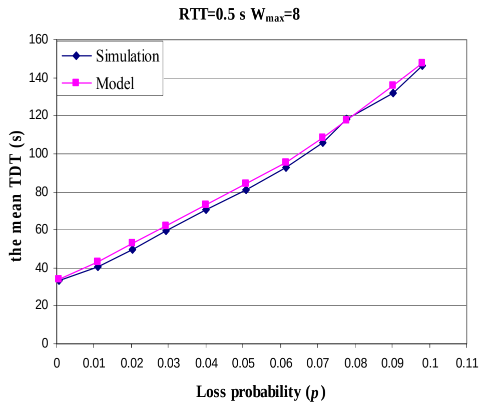












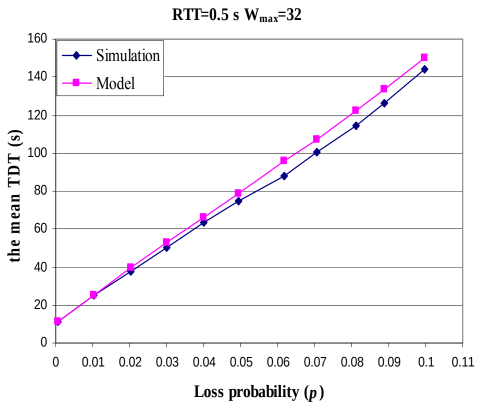


Figure 6-3: Comparison between the estimated and simulated TDT values in a multi-domain environment in the case that packet loss occurs only on the link from the sender to the receiver

	RTT = 0.06 s, Wmax = 8			RTT = 0.06 s, Wmax = 32		
p	NS	Multi-domain TDT model		NS	Multi-domain TDT model	
	the mean TDT	the mean TDT	ε_{rel}	the mean TDT	the mean TDT	ε_{rel}
0.0006	4.12	4.14	**1%**	1.40	1.36	**-3%**
0.0106	5.10	5.21	**2%**	3.25	3.14	**-3%**
0.0206	6.23	6.47	**4%**	4.90	4.89	**0%**
0.0306	7.46	7.68	**3%**	6.47	6.60	**2%**
0.0406	9.16	8.93	**-3%**	7.85	8.14	**4%**
0.0506	10.36	10.22	**-1%**	9.63	9.78	**2%**
0.0606	11.86	11.33	**-4%**	11.11	11.49	**3%**
0.0706	13.61	13.07	**-4%**	12.80	13.01	**2%**
0.0806	15.23	14.70	**-3%**	14.59	14.76	**1%**
0.0905	16.75	16.57	**-1%**	16.26	16.60	**2%**
0.1005	18.82	18.35	**-3%**	18.29	18.49	**1%**

	RTT = 0.1 s, Wmax = 8			RTT = 0.1 s, Wmax = 32		
p	NS	Multi-domain TDT model		NS	Multi-domain TDT model	
	the mean TDT	the mean TDT	ε_{rel}	the mean TDT	the mean TDT	ε_{rel}
0.0006	6.78	6.84	**1%**	2.29	2.25	**-2%**
0.0106	8.35	8.58	**3%**	5.10	5.16	**1%**
0.0206	10.13	10.72	**6%**	7.88	7.98	**1%**
0.0306	12.30	12.71	**3%**	10.15	10.87	**7%**
0.0406	14.64	14.71	**0%**	12.94	13.47	**4%**
0.0506	16.55	16.94	**2%**	15.43	15.91	**3%**
0.0606	19.29	18.90	**-2%**	18.44	19.03	**3%**
0.0706	22.24	21.58	**-3%**	21.21	21.67	**2%**
0.0806	24.43	24.21	**-1%**	23.62	24.59	**4%**
0.0905	27.83	27.28	**-2%**	26.22	27.37	**4%**
0.1005	30.81	30.32	**-2%**	29.60	30.30	**2%**

	RTT = 0.3 s, Wmax = 8			RTT = 0.3 s, Wmax = 32		
p	NS	Multi-domain TDT model		NS	Multi-domain TDT model	
	the mean TDT	the mean TDT	ε_{rel}	the mean TDT	the mean TDT	ε_{rel}
0.0006	20.16	20.33	**1%**	6.76	6.68	**-1%**
0.0106	24.73	25.79	**4%**	14.83	15.40	**4%**
0.0206	30.01	31.78	**6%**	22.83	24.05	**5%**
0.0306	35.66	37.72	**6%**	30.17	32.11	**6%**
0.0406	42.30	44.05	**4%**	38.03	39.52	**4%**
0.0506	48.40	50.45	**4%**	45.30	47.44	**5%**
0.0606	55.34	55.49	**0%**	54.08	56.87	**5%**
0.0706	62.94	64.35	**2%**	61.37	64.32	**5%**
0.0806	70.67	71.20	**1%**	69.03	72.62	**5%**
0.0905	80.03	81.44	**2%**	78.52	80.52	**3%**
0.1005	89.64	89.05	**-1%**	87.23	91.06	**4%**

	RTT = 0.5 s, Wmax = 8			RTT = 0.5 s, Wmax = 32		
p	NS	Multi-domain TDT model		NS	Multi-domain TDT model	
	the mean TDT	the mean TDT	ε_{rel}	the mean TDT	the mean TDT	ε_{rel}
0.0006	33.41	33.83	**1%**	11.24	11.11	**-1%**
0.0106	40.64	43.20	**6%**	25.43	25.26	**-1%**
0.0206	49.86	52.76	**6%**	37.96	39.92	**5%**
0.0306	59.54	61.77	**4%**	50.14	53.13	**6%**
0.0406	70.58	72.96	**3%**	63.25	66.00	**4%**
0.0506	80.72	83.92	**4%**	74.51	78.93	**6%**
0.0606	92.64	95.29	**3%**	87.63	95.86	**9%**
0.0706	105.60	108.53	**3%**	100.19	107.35	**7%**
0.0806	118.30	117.80	**0%**	114.25	122.46	**7%**
0.0905	131.81	135.73	**3%**	126.52	133.67	**6%**
0.1005	146.41	147.36	**1%**	144.18	149.96	**4%**

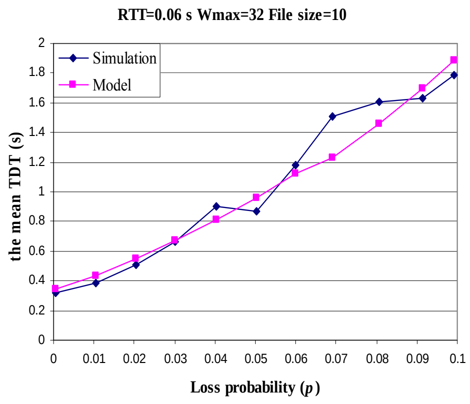
Table 6-1: Numerical results of the estimated and the simulated mean TDT in seconds

In the case where packets are only lost in the direction from the sender to the receiver, Figure 6-3 shows that our model for predicting the mean TDT in a multi-domain environment matches quite well with the simulated mean TDT values. In Table 6-1, we notice that all relative errors of the model are within ±9%. Most of relative errors are within ±6%. To confirm the accuracy of the model for predicting the mean TDT in a multi-domain environment, more experiments are executed. In these experiments, some of the parameters are modified. The modified parameters are listed below, while the remaining parameters are unaltered.

- W_{max}: 32 packets
- Minimum RTO: 1 second
- File size: 10 packets
- p_{f_3} : 0.0005, p_{b_3} : 0, RTT_3 : $\{0.02, 0.46\}s$, B_3 : $1Gbps$

Based on these new parameter settings, we obtained the simulation results by performing 30 independent simulation runs in which 20 servers are active to send a file with 10 packets. In each run the average TDT of the 20 servers is computed and finally the average TDT of the 30 individual runs are taken for comparison with the TDT estimated by the model. The comparison results are shown in Figure 6-4 and Table 6-2. Obviously, for these new scenarios, the model performs a little worse. But it is understandable. The Cardwell TDT model is based on the goodput model of Padhye that assumes that there are large amounts of data to send. Therefore, there is no doubt that based on this goodput, the model for predicting the TDT of a small file is less accurate than that for predicting the TDT of a file with a large file size. But in Table 6-2, we notice that mostly the relative errors are within ±10%. This result is not bad. Therefore, we ensure the accuracy of the model in Section (6.1) for predicting the mean TDT in a multi-domain environment when packet loss occurs only on the link from the sender to the receiver.

From the right graph of Figure 6-4, we notice an interesting phenomenon. The mean TDT predicted by the model for $p = 0.07$ is lower than that for $p = 0.06$. Initially, we suppose that for the same RTT, a higher loss probability should cause the longer duration of downloading a file. But that is not always the case. From (5.4), we obtain that the expected number of data packets sent in the initial slow start for $p = 0.06$ is 7.2 packets and for $p = 0.07$ that is equal to 6.85 packets. According to the SS mechanism described in Section 2.2.2.1, we obtain that the time spent in the initial SS for $p = 0.06$ and $p = 0.07$ is $4RTT$ and $3RTT$ respectively. After the initial SS phase, the number of remaining data packets for those two cases are 2.8 and 3.15 packets respectively. To detect packet loss by triple DA, the remaining number of data packets must be at least 3 packets. Therefore for $p = 0.06$, the lost packet can only be detected by a RTO. For $p = 0.07$, that is not necessarily the case. Hence, the time taken to recovery the lost packet for $p = 0.06$ is longer than that for $p = 0.07$. Finally the time taken to send the remaining data for both cases is nearly equal to each other. Both of them can be approximated as $2RTT$. After the loss is recovered, for $p = 0.06$, it enters into the SS phase with the initial window size 1. Thus it takes 2 rounds to send the remaining 2.8 packets. For $p = 0.07$, it enters mostly in the CA phase. Then the *cwnd* is half of the window size when the first loss occurs. It can be verified that the *cwnd* is equal to 2 packets. Therefore, it also takes two rounds to send the remaining data. Combining all the results mentioned above, we can explain why the mean TDT predicted by the model for $p = 0.07$ is lower than that for $p = 0.06$.



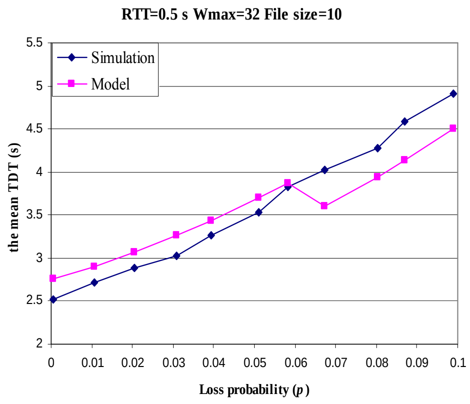


Figure 6-4: Comparison between the model and the simulation for the new scenarios

	RTT = 0.06 s, Wmax = 32			RTT = 0.5 s, Wmax = 32		
p	NS	Multi-domain TDT model		NS	Multi-domain TDT model	
	the mean TDT	the mean TDT	ε_{rel}	the mean TDT	the mean TDT	ε_{rel}
0.0006	0.32	0.34	**7%**	2.52	2.77	**10%**
0.0106	0.39	0.44	**14%**	2.71	2.90	**7%**
0.0206	0.51	0.55	**7%**	2.89	3.07	**6%**
0.0306	0.66	0.67	**1%**	3.03	3.26	**8%**
0.0406	0.90	0.81	**-10%**	3.26	3.44	**6%**
0.0506	0.87	0.96	**10%**	3.53	3.70	**5%**
0.0606	1.18	1.12	**-5%**	3.83	3.87	**1%**
0.0706	1.50	1.23	**-18%**	4.03	3.60	**-11%**
0.0806	1.60	1.46	**-9%**	4.28	3.95	**-8%**
0.0905	1.63	1.70	**4%**	4.59	4.14	**-10%**
0.1005	1.79	1.88	**5%**	4.91	4.50	**-8%**

Table 6-2: Numerical results of the estimated and simulated mean TDT in unit of second

6.2.2 *Model validation for the case when a packet loss occurs on two ways*

In this section we validate the mean TDT model in a multi-domain environment in the case that not only data packets experience loss but also the ACKs of the received data. First, we list the parameter settings below.

- MSS: 1640 Bytes
- W_{max}: {8, 32} packets
- b: the number of packets that is acknowledged by each ACK: 1
- w_1: the initial window size: 1
- Minimum RTO: 4 times the *RTT* of the entire network

- File size: 500 packets

In the four networks we have the following assumptions:

Network 1

$p_{f_1} = p_{b_1} = 0,\ RTT_1 : 0.02s,\ B_1 : 10Mbps$

Network 2

$p_{f_2} = p_{b_2} = 0.0001,\ RTT_2 : 0.01s,\ B_2 : 1Gbps$

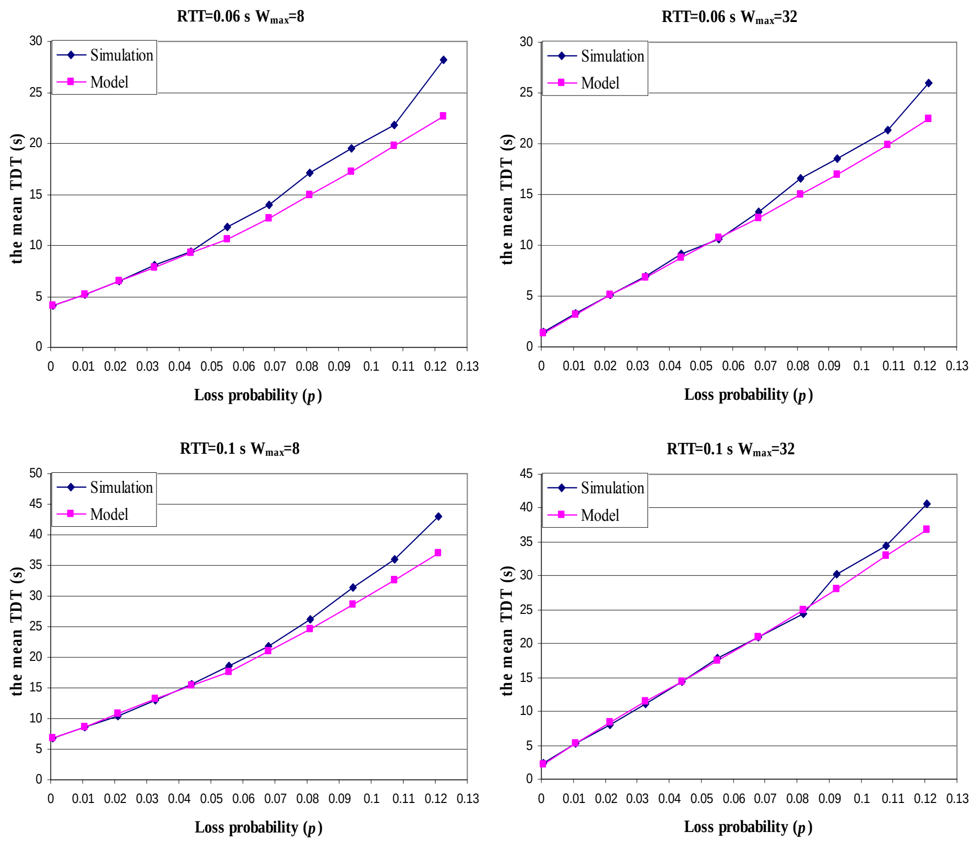
Network 3

$p_{f_3} = p_{b_3} = 0.0005,\ RTT_3 : \{0.02,\ 0.06,\ 0.26, 0.46\}s,\ B_3 : 1Gbps$

Network 4

$p_{f_4} = p_{b_4} = \{0,\ 0.01,\ ...,\ 0.1\},\ RTT_4 :\ \ 0.01s,\ B_4 : 1Gbps$

The estimated and simulated mean TDT are both expressed in the unit of seconds. The comparison results of the estimated and the simulated TDT are shown in Figure 6-5 and in Table 6-3. The *x*-axis in Figure 6-5 represents the probability *p,* which is defined as the probability that a packet is considered lost by the sender TCP. The first column of Table 6-3 contains the forward and backward packet loss probabilities in the entire network. For the parameter settings above, Figure 6-5 and Table 6-3 demonstrate that the model discussed in Section 6.1 is accurate for predicting the mean TDT in a multi-domain environment when a packet loss occurs in the direction from the sender to the receiver and also in the reverse direction.



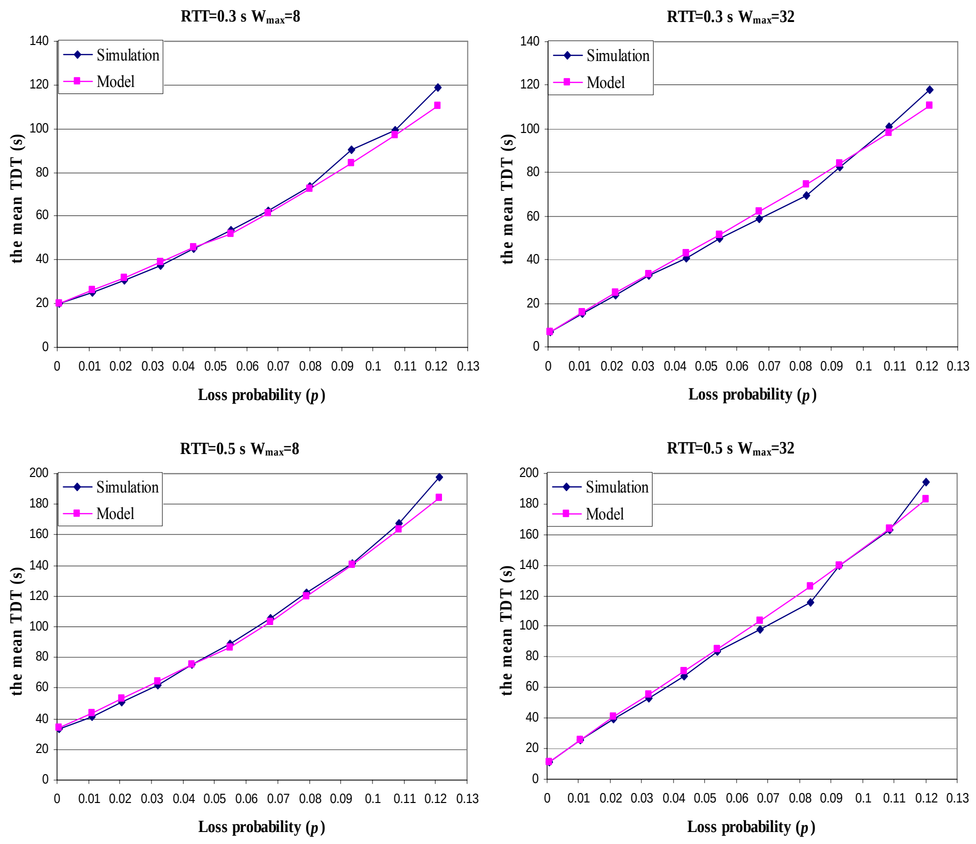


Figure 6-5: Comparison between estimated and simulated mean TDT values in a multi-domain environment when packet losses occur in the direction from the sender to the receiver and in the reverse direction

	RTT = 0.06 s, Wmax = 8			RTT = 0.06 s, Wmax = 32		
$p_f = p_b$	NS	Multi-domain TDT model		NS	Multi-domain TDT model	
	the mean TDT	the mean TDT	ε_{rel}	the mean TDT	the mean TDT	ε_{rel}
0	4.13	4.15	**0%**	1.40	1.36	**-3%**
0.01	5.23	5.20	**-1%**	3.29	3.17	**-4%**
0.02	6.47	6.54	**1%**	5.07	5.06	**0%**
0.03	8.01	7.88	**-2%**	6.95	6.85	**-2%**
0.04	9.44	9.34	**-1%**	9.13	8.73	**-4%**
0.05	11.80	10.58	**-10%**	10.56	10.68	**1%**
0.06	13.97	12.65	**-9%**	13.29	12.67	**-5%**
0.07	17.08	14.89	**-13%**	16.57	14.99	**-10%**
0.08	19.55	17.17	**-12%**	18.52	17.00	**-8%**
0.09	21.84	19.71	**-10%**	21.38	19.87	**-7%**
0.1	28.20	22.59	**-20%**	25.94	22.39	**-14%**

	RTT = 0.1 s, Wmax = 8			RTT = 0.1 s, Wmax = 32		
p_f=p_b	NS	Multi-domain TDT model		NS	Multi-domain TDT model	
	the mean TDT	the mean TDT	ε_{rel}	the mean TDT	the mean TDT	ε_{rel}
0	6.78	6.84	**1%**	2.35	2.25	**-4%**
0.01	8.50	8.64	**2%**	5.37	5.21	**-3%**
0.02	10.37	10.74	**4%**	8.08	8.35	**3%**
0.03	13.03	13.15	**1%**	11.06	11.39	**3%**
0.04	15.57	15.48	**-1%**	14.43	14.46	**0%**
0.05	18.66	17.65	**-5%**	17.84	17.47	**-2%**
0.06	21.77	20.94	**-4%**	20.87	20.96	**0%**
0.07	26.19	24.68	**-6%**	24.50	25.05	**2%**
0.08	31.47	28.60	**-9%**	30.29	28.03	**-7%**
0.09	36.03	32.62	**-9%**	34.45	32.92	**-4%**
0.1	42.96	37.01	**-14%**	40.66	36.82	**-9%**
	RTT = 0.3 s, Wmax = 8			RTT = 0.3 s, Wmax = 32		
p_f=p_b	NS	Multi-domain TDT model		NS	Multi-domain TDT model	
	the mean TDT	the mean TDT	ε_{rel}	the mean TDT	the mean TDT	ε_{rel}
0	20.14	20.34	**1%**	6.82	6.68	**-2%**
0.01	25.00	25.94	**4%**	15.52	15.57	**0%**
0.02	30.66	32.03	**4%**	23.94	25.03	**5%**
0.03	37.13	39.10	**5%**	32.53	33.39	**3%**
0.04	45.44	45.59	**0%**	40.67	42.89	**5%**
0.05	53.64	51.98	**-3%**	49.68	51.47	**4%**
0.06	62.44	61.34	**-2%**	58.97	61.83	**5%**
0.07	73.50	72.75	**-1%**	69.64	74.64	**7%**
0.08	90.48	84.13	**-7%**	82.40	83.95	**2%**
0.09	99.09	97.13	**-2%**	101.25	98.06	**-3%**
0.1	118.68	110.55	**-7%**	118.08	110.68	**-6%**
	RTT = 0.5 s, Wmax = 8			RTT = 0.5 s, Wmax = 32		
p_f=p_b	NS	Multi-domain TDT model		NS	Multi-domain TDT model	
	the mean TDT	the mean TDT	ε_{rel}	the mean TDT	the mean TDT	ε_{rel}
0	33.48	33.83	**1%**	11.28	11.11	**-2%**
0.01	41.14	43.43	**6%**	25.34	25.76	**2%**
0.02	50.81	53.18	**5%**	39.56	41.08	**4%**
0.03	62.22	64.44	**4%**	52.73	55.72	**6%**
0.04	75.14	75.48	**0%**	67.72	70.75	**4%**
0.05	88.95	86.45	**-3%**	83.29	85.42	**3%**
0.06	105.91	103.48	**-2%**	98.24	103.43	**5%**
0.07	122.09	119.64	**-2%**	115.60	126.25	**9%**
0.08	141.39	140.89	**0%**	139.52	139.62	**0%**
0.09	167.83	163.86	**-2%**	162.93	163.90	**1%**
0.1	197.87	184.24	**-7%**	194.00	182.96	**-6%**

Table 6-3: Numerical results of the mean TDT in a multi-domain environment for the case that loss happens in both directions on the network links

As in Section 6.2.1, we investigate further if the model performs well under other scenarios. In the new experiments, we modified some parameters and list them below. For the TCP parameters we choose the following values:

- Maximum segment size (*MSS*) : 1640 Bytes
- Maximum advertised window size (W_{max}): 32 packets
- The number of data packets acknowledged by one ACK (*b*): 1
- The initial SS window size (w_1): 1 packet
- Minimum RTO: 2 seconds
- File size: 10 packets

In the four networks we have the following assumptions:

Network 1:

$p_{f_1} = p_{b_1} = 0,\ RTT_1 : 0.02s,\ B_1 : 10Mbps$

Network 2

$p_{f_2} = p_{b_2} = 0.0001,\ RTT_2 : 0.01s,\ B_2 : 1Gbps$

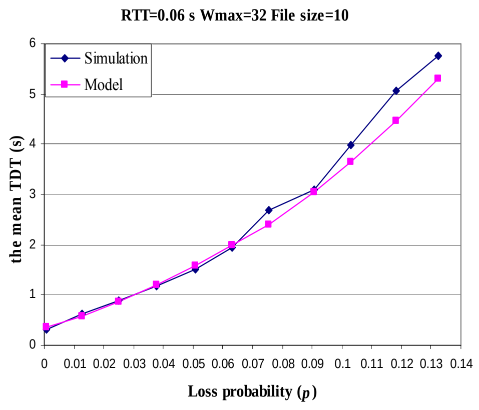
Network 3

$p_{f_3} = p_{b_3} = 0.0005,\ RTT_3 : \{0.02, 0.06, 0.26, 0.46\}s,\ B_3 : 1Gbps$

Network 4

$p_{f_4} = p_{b_4} = \{0, 0.01, ..., 0.1\},\ RTT_4 :\ 0.01s,\ B_4 : 1Gbps$

Based on these new parameter settings, the simulation results are gained by performing 10 independent simulation runs in which 150 servers are active to send the file with 10 packets. In each run the average TDT of the 150 servers are computed and finally the average TDT of the 10 individual runs are taken to compare the estimated TDT. The comparison results are shown in Figure 6-6 and Table 6-4.



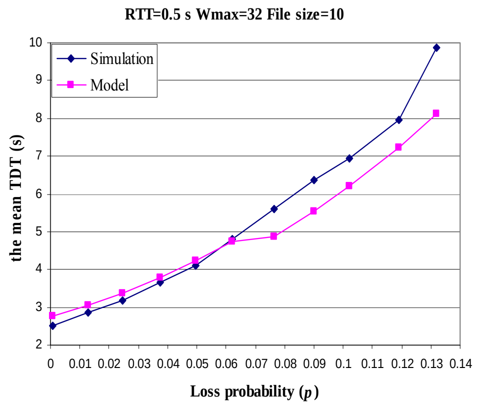


Figure 6-6: Comparison between the model and the simulation under the new parameter settings

	RTT = 0.06 s, Wmax = 32			RTT = 0.5 s, Wmax = 32		
p_f=p_b	NS	Multi-domain TDT model		NS	Multi-domain TDT model	
	the mean TDT	the mean TDT	ε_{rel}	the mean TDT	the mean TDT	ε_{rel}
0	0.32	0.35	**10%**	2.53	2.77	**10%**
0.01	0.61	0.58	**-6%**	2.84	3.05	**7%**
0.02	0.89	0.86	**-3%**	3.19	3.38	**6%**
0.03	1.16	1.19	**2%**	3.66	3.79	**4%**
0.04	1.52	1.58	**4%**	4.11	4.23	**3%**
0.05	1.96	2.00	**2%**	4.80	4.73	**-1%**
0.06	2.68	2.39	**-11%**	5.60	4.86	**-13%**
0.07	3.09	3.05	**-1%**	6.38	5.54	**-13%**
0.08	3.99	3.64	**-9%**	6.95	6.21	**-11%**
0.09	5.07	4.45	**-12%**	7.96	7.23	**-9%**
0.1	5.76	5.31	**-8%**	9.88	8.11	**-18%**

Table 6-4: Numerical results of the estimated and simulated TDT under the new parameter settings

The figure and the table above demonstrate that the outcomes of the simulation and the model are quite similar. Therefore, we conclude that the model described in Section 6.1 is accurate for predicting the mean TDT in a multi-domain environment.

7 Correlated loss pattern

In the previous chapters, it is shown that the TDT models are very accurate for predicting the mean TDT in a single- and multi-domain environment when packet losses are uncorrelated. In this chapter, we discuss how to apply those models for predicting the mean TDT when packet losses are correlated. We will assume that packet losses occur according to the two-state Gilbert loss model. For simplification, we just focus on the Cardwell TDT model described in Section 5.1. The application of that model to predict the TDT in the case that packet loss follows the Gilbert loss model is described in Section 7.1. The validation is described in Section 7.2. Finally, a conclusion is derived in Section 7.3.

7.1 Two-state Gilbert loss model

In this section we discuss the case where packet losses are correlated. According to [27], the correlation structure of the packet loss process can be modeled with low order Markov chains. In particular, the two-state Gilbert model was found to be an accurate model in many studies. Therefore we use the Gilbert model to simulate packet loss patterns over links.
First we describe the two-state Gilbert model depicted in Figure 7-1. In the Gilbert model, one state represents a packet loss, which is called state '*B*', and the other state represents the situation when a packet is successfully delivered to the destination which is referred to as state '*G*'. *m* is the probability of going from state '*G*' to state '*B*', and *n* is that of staying in state 'B'. Note when *m* is equal to *n*, packet loss is uncorrelated. In addition, when *n* is larger than *m*, packet loss is bursty.

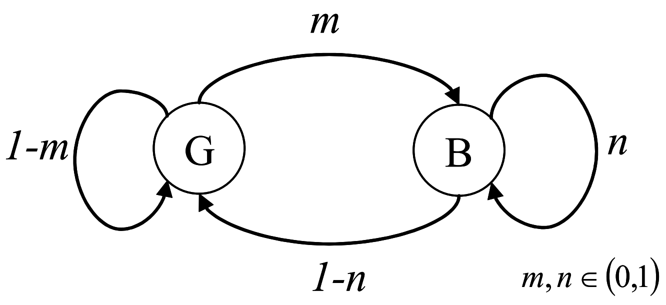


Figure 7-1: The two-state Gilbert loss model

On the link between the sender to the receiver, we denote π_G as the probability that a packet is delivered successfully and π_B as the average packet drop rate. It is easy to verify that these two probabilities satisfy the following equations:

$$\pi_G = (1-m)\pi_G + (1-n)\pi_B, \tag{7.1}$$

$$\pi_G + \pi_B = 1. \tag{7.2}$$

From (7.1) and (7.2), we obtain

$$\pi_G = \frac{1-n}{1+m-n}, \tag{7.3}$$

and

$$\pi_B = \frac{m}{1+m-n}. \tag{7.4}$$

Substituting the packet loss probability defined in the Cardwell TDT model p by π_B, we get the model for predicting the mean TDT in the situation when packet losses are correlated and simulated in the two-state Gilbert loss model.

7.2 Numerical results

In this section we validate the model discussed in the previous section by ns-2 simulations. In these simulation experiments, we reuse the simulation topology depicted in Figure 3-7. As depicted in Figure 3-7, we append an error model on the link from the router and the receiver. The error module we choose to simulate the Gilbert loss model is *ErrorModel/TwoState*. This module is described in [9]. For simplification, packet losses in the reverse direction are ignored.

For the TCP parameters we choose the following values for the experiments.

- Maximum segment size (*MSS*) : 1000 Bytes
- Maximum advertised window size (W_{max}): 32 packets
- The number of data packets acknowledged by one ACK (*b*): 1
- The initial SS window size (w_1): 1 packet
- Access-link capacity: 10 Mbps
- Bottleneck-link capacity: 200 Mbps
- Fixed round-trip time (*RTT*): 0.2 seconds
- Probability of going from state '*G*' to state '*B*' (*m*): {0, 0.01, …, 0.1}
- Probability of going from state '*B*' to state '*G*' (*n*): $\{m/5, 5m\}$
- File size: 500 packets
- Minimum RTO: $4RTT$

The comparison outcomes of the mean TDT obtained from the simulation and predicted by the model are provided in Figure 7-2 and in Table 7-1. The mean TDT is in the unit of seconds. The *x*-axis in Figure 7-2 and the first column in Table 7-1 represent *m*, denoted as the probability of going from state '*G*' to state '*B*'. The figure and the table show that the TDT outcomes obtained from the simulation is quite close to the TDT results predicted by the model. In addition, for $n = 5m$, the mean TDT increases much faster than for $n = m/5$.

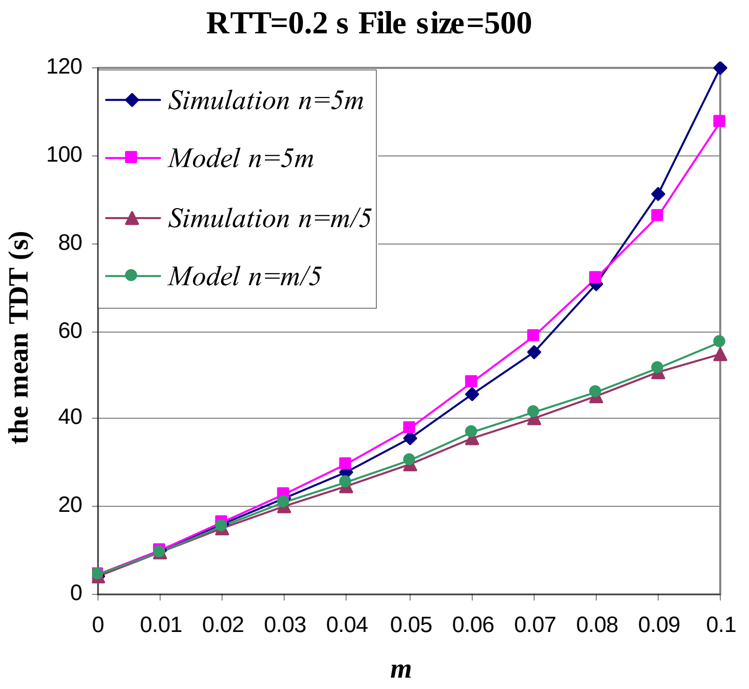


Figure 7-2: Comparison of the model and the simulation for the two-state Gilbert model

	RTT = 0.2 s, $q = p/5$			RTT = 0.2 s, $q = 5p$		
p	NS	Multi-domain TDT model		NS	Multi-domain TDT model	
	the mean TDT	the mean TDT	ε_{rel}	the mean TDT	the mean TDT	ε_{rel}
0	4.23	4.36	**3%**	4.23	4.37	**3%**
0.01	9.61	9.59	**0%**	9.88	9.92	**0%**
0.02	15.05	15.46	**3%**	15.75	16.28	**3%**
0.03	20.01	20.82	**4%**	21.73	22.84	**5%**
0.04	24.79	25.37	**2%**	27.93	29.62	**6%**
0.05	29.57	30.62	**4%**	35.74	37.79	**6%**
0.06	35.68	37.11	**4%**	45.80	48.48	**6%**
0.07	39.98	41.38	**4%**	55.14	58.87	**7%**
0.08	45.34	46.09	**2%**	70.95	72.10	**2%**
0.09	50.59	51.78	**2%**	91.34	86.36	**-5%**
0.1	54.74	57.50	**5%**	119.91	107.79	**-10%**

Table 7-1: Numerical results of the estimated and simulated for the two-state Gilbert loss model

7.3 Conclusion

In Section 7.1, we discussed the application of the Cardwell TDT model for predicting the TDT in the case that packet losses follow the two-state Gilbert model. Based on the validation in Section 7.2, we conclude that when packet loss on links is correlated and the correlation of packet loss is known, we can apply the Cardwell model to predict the mean TDT by substituting the packet loss probability p by π_B. This application holds also for the uncorrelated packet loss case. Actually, uncorrelated packet loss with loss probability δ can be modeled by setting m and n in the Gilbert model to δ. Then from (7.4), we obtain that π_B is also equal to δ.

8 Summary and topics for further research

8.1 Summary

In this research, we first extended the well-known Padhye throughput model for TCP by taking the behaviour of TCP's slow start and the fast recovery mechanism into account. Secondly, we developed a model for predicting the Response Time (RT) in a single- and multi-domain environment. Then we extended the Cardwell Total Download Time (TDT) model for predicting the mean TDT in a single- and multi-domain environment by considering packet losses in the direction from the server to the client and in the reverse direction. Finally, we investigated whether the TDT model performs well to predict the mean TDT when packet losses are correlated and the loss pattern is simulated according to the two-state Gilbert loss model. The conclusions we can draw from these investigations are stated below.

The extended TCP throughput model

- In the case that window limitation has no impact, the extended model is in general more accurate than the Padhye throughput model.
- If window limitation has impact, the extended model performs a little worse for the loss probabilities lower than 3%, but for probabilities larger than 3%, the extended model performs much better.

The mean RT model in a single- and multi-domain environment

- We conclude from various simulations, wherein the packet loss probability has been varied between 0-16%, that our RT model is very accurate for predicting the mean RT in a single- and multi-domain environment. In fact, the analytical results for the mean RT always fall within the 95% confidence interval of the simulated values.

The extended Cardwell model in a single- and multi-domain environment

- For the forward and the backward packet loss probability varied between 0-10%, the extended Cardwell model works very well for predicting the mean TDT for large files. The relative errors of the model are mostly between -5% and +5%.
- For predicting the mean TDT of a small file (10 packets), the extended model is accurate under the condition that the forward and the backward loss probability are not higher than 5%. Under this condition, the relative errors of the model are between -10% and +10%.

The applied Cardwell model to predict the mean TDT for correlated packet losses

- If correlated packet losses are simulated according to the two-state Gilbert loss model, the Cardwell TDT model is very accurate for predicting the mean TDT by substituting the packet loss probability *p*, by π_B (the average packet drop rate, see Section 7.4).

8.2 Topics for further research

In this project, the extended Padhye model, the RT model, and the TDT model are validated by ns-2 simulations. By comparing the estimated and simulated outcomes demonstrated in the previous chapters, we conclude that these models are accurate for predicting the TCP throughput, the response time, and the total download time. However, while we have considerable confidence in ns-2 simulations, ns-2 is not a polished and finished product [9]. Although it attempts to model real world networks as good as possible, it will not be perfect [28]. Therefore, we recommend to further validate the models by real measurements.

To apply the models for predicting the mean TDT, we have to be able to estimate the value of T_0, denoting as the average duration of the first time-out in a sequence of one or more successive time outs. In our work, we assume that the *RTT* is constant. Under this assumption, we obtained Equation (2.7) for predicting the time-out after *N* updates. It is easy to verify that the maximum value of (2.7) is equal to $3RTT$. To reduce the complexity of determining T_0, we set in our simulation the minimum RTO to at least $4RTT$. Therefore T_0 can be estimated by the minimum RTO. However, the actual *RTT* in a dynamic environment may vary over time. T_0 must be consequently updated. In addition, in the case that the minimum RTO is a small value, this estimation will lead to inaccuracy. Therefore we recommend developing a new model for predicting the T_0 in a dynamic environment.

Finally, we propose to develop a model for predicting the distribution of the RT and the TDT. In this project, we only focus on the mean RT and the mean TDT. In real life, the RT and the TDT are variable. Hence, finding the distribution of the RT and the TDT is an interesting topic.

Bibliography

[1] University of Waikato Computer Science Department, *Waikato Internet Traffic Storage*, http://wand.cs.waikato.ac.nz/wand/wits/auck/6/

[2] University of Wisconsin Madison Division of Information Technology, Network
Performance Statistics, http://wwwstats.net.wisc.edu/

[3] Cooperative Association for Internet Data Analysis, Traffic Workload verview, http://www.caida.org/outreach/resources/learn/trafficworkload/tcpudp.xml

[4] EQUANET, *End-to-End Quality of Service in Next-Generation Networks*, http://equanet.cs.utwente.nl/

[5] J.Padhye, V.Firoiu, D. Towsley, J.Kurose, Modeling TCP throughput: A Simple
Model and its Empirical Validation, IEEE/ACM Transactions on Networking, 8(2): 133-145 (April), 2000

[6] Matthew Mathis, Jeffrey Semke, Jamshid Mahdavi, *The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm*, Computer Communication Review, 27(3), July 1997

[7] J. Mahdavi and S. Floyd, *TCP-friendly unicast rate-based flow control*, Note sent to end2end-interest mailing list, Jan 1997

[8] T. Ott, J. Kemperman, and M. Mathis, *The stationary behavior of ideal TCP congestion avoidance*. in preprint

[9] Network Simulator 2 (software), http://www.isi.edu/nsnam/ns

[10] J. Padhye, V. Firoiu, D. Towsley, *A stochastic model of TCP Reno congestion avoidance and control*, Technical Report 99-02, University of Massachussets, 1999

[11] N. Cardwell, S. Savage, T. Anderson, Modeling TCP latency, Proceedings of INFOCOM 2000, March 2000

[12] W. R. Stevens, *TCP/IP Illustrated Volume I: The Protocols*, Addison-Wesley, 1994

[13] J. Casad, *Sams Teach Yourself TCP/IP in 24 Hours, 3rd Edition*, Sams, 2003

[14] Jacobson, V., *Congestion Avoidance and Control*, Computer Communication Review, vol. 18, no. 4, pp. 314-329, Aug 1988

[15] Jacobson, V., *Modified TCP Congestion Avoidance Algorithm*, Note sent to end2end-interest mailing list, April 30, 1990

[16] W. Stevens, TCP slow start, congestion avoidance, fast retransmit, and fast recovery algorithms, Internet Archives RFC 2001, January 1997

[17] B. van leeuwen, *A short introduction to TCP*

[18] V.Paxson, M.Allman, Computing TCP's Retransmission Timer, Internet Archives RFC 2988, November 2000

[19] S. Floyd, M. Handley, J. Padhye, J. Widmer, Equation-Based Congestion Control for Unicast Applications, ACM SIGCOMM, 2000

[20] H. Ekström, R. Ludwig, *The Peak-Hopper: A New End-to-End Retransmission Timer for Reliable Unicast Transport*, IEEE INFOCOM, 2004

[21] M. Allman, V. Paxon, *On Estimating End-to-End Network Path Properties*, ACM SIGCOMM, 1999

[22] S. Jacobsson, "Performance Evaluation of the Eifel Retransmission Timer", Master's Thesis, Aachen University of Technology, October 2001

[23] Y. Lee, *Introduction to TCP/IP*, http://www.hep.ucl.ac.uk/~ytl/tcpip/background/tahoe-reno.html

[24] M. Hassan, R. Jain. *High performance TCP/IP networking Concepts, Issues and*
Solutions, Prentice-Hall, 2003

[25] K. Ahmed, *Perceived web browsing quality - A literature survey*, EQAUNET projectnumber TSIT2031, 2003

[26] *Maple 9 (software)*, http://www.maplesoft.com/

[27] C. Boutremans J.Y. Le Boudec, *Adaptive Joint Playout Buffer and FEC Adjustement for Internet Telephony*, IEEE INFOCOM, 2003

[28] OPENXTRA, *An Introduction to Network Simulation*, http://www.openxtra.co.uk/articles/network-simulation.php