Reliable Network Transmission Protocol Modeling and Design

By

Dabin Wang

A thesis
Submitted to the Faculty of Graduate Studies
in Partial Fulfillment of the Requirements
for the Degree of

MASTER OF SCIENCE

Department of Electrical and Computer Engineering
University of Manitoba
Winnipeg, Manitoba

December, 1998

THE UNIVERSITY OF MANITOBA

FACULTY OF GRADUATE STUDIES

*****

COPYRIGHT PERMISSION PAGE

RELIABLE NETWORK TRANSMISSION PROTOCOL MODELING AND DESIGN

BY

DABIN WANG

A Thesis/Practicum submitted to the Faculty of Graduate Studies of The University of Manitoba in partial fulfillment of the requirements of the degree of

MASTER OF SCIENCE

To my parents, my wife, and my son

for their love, support, and understanding

Abstract

In this thesis we cover two main topics, reliable network transmission protocol modeling and design. The first is a reliable transmission protocol modeling by the place stochastic Petri net. The initial idea was derived from timing analysis for synthesis in microprocessor interface design which was introduced by Marco A. Escalante and Nikitas J. Dimopoulos at the University of Victoria. The second is retransmission timer design for TCP. The first consideration leads to a pursuit of the tightest bounds of the timing constraints given in a specification using functional optimization. To illustrate use of the technique, we investigated the deterministic timeout design in TCP and discussed delay buffer design in the real-time communication in the Internet.

For TCP, this leads to a requirement for traffic measurements and observations that resulted in two main results. First, we found the packet round trip time over IP has the self-similar property. To our knowledge, no one has found and considered this property about the round-trip time of transmmitted packets over IP. Second, we attempted to estimate the parameters in Jacobson's dynamic retransmission timeout algorithm by making use of self-similarity parameter.

We also have other obsearvations such as a relationship between the loss rate and Hurst parameter. We also found that our modeling is well suited to the design of a packet delay buffer with a given reliability factor in a real-time application in the Internet.

In the recent years, many researchers reported that actual network traffic is self-similar in nature. However, effectively designing protocols that take self-similarity into account remains largely an open issue. There are two major problems: (1) What is the physical "explanation" for observed self-similar nature of measured traffic from today's packet networks? (2) What is the impact of self-similarity on network and protocol design and performance analysis? Walter Willinger, Murad S. Taqqu, Robert Sherman, and Daniel V. Wilson and related studies answered question (1). The study here is related to question (2).

Acknowledgements

I wish to thank my supervisor Dr. R. D. McLeod for his help and encouragement while I was working on the project and writing this work. Special thanks go to my wife, Sihong Lei. I gratefully acknowledge the partial financial support from Dr. R. D. McLeod and Dr. D. C. Blight through the Micronet Centre of Excellence. I would also like to thank Mr. Kent Felske at Nortel Telecom, Ottawa. Finally, I express my sincere thanks to my friends for their help.

Contents

Abstract (i)

Acknowledgments (ii)

Chapter 1 Introduction 1

Chapter 2 Concepts of Petri Net 5
2.1 Ordinary Petri Nets 5
2.1.1 Transition Enabling and Firing 5
2.1.2 Behavioural Properties 7
2.1.3 Methods of Analysis 9
2.2 Stochastic Petri Net 12

Chapter 3 Overview of Transport Protocol: TCP 13
3.1 TCP/IP Protocol Suite 13
3.2 Transmission Control Protocol 16
3.2.1 TCP Flow Control 17
3.2.2 TCP Congestion Control 18

Chapter 4 A Probabilistic Timing Analysis for Synthesis in Reliable Transmission Design 20
4.1 Timing Analysis 21
4.1.1 Traditional Stochastic Petri Net 21
4.1.2 Place Stochastic Petri net 21
4.1.3 Computation of the Delay Between Transitions 24
4.2 Timing Analysis for Synthesis 25
4.3 Distribution of Round Trip Time and Deterministic Timeout Design 29
4.4 Application to Real-time Communication 32
4.5 Summary 35

Chapter 5 Self-Similarity of the Round Trip Time and the Jacobson's Retansmission Algorithm 36
5.1 Introduction 36
5.2 The Self-Similarity Property 38
5.3 Parameters of Jacobson Algorithm and the Hurst Parameter 42
5.4 Examples 45
5.5 Summary 48

Chapter 6 Other Observations and Discussions 49
6.1 Loss Rate and Hurst Parameter 49
6.2 Estimation of the Number of Nodes 50
6.2.1 Theory 50
6.2.2 Testing in the Internet 51
6.3 On 24-hour Statistical Data of Loss Rate 55

Chapter 7 Conclusion 56

Appendix 58

References 61

Chapter 1

Introduction

The aim of this research is reliable network transmission modeling and design. The original idea came from the timing analysis for synthesis in microprocessor interface design, introduced by Marco A. Escalante and Nikitas J. Dimopoulos [6, 7, 8]. Basic concepts are overviewed in Chapter 2 and Chapter 3. To illustrate our use of the technique, we investigated the deterministic timeout design in TCP and a delay buffer design in the real-time network. For TCP, this required traffic measurements and observations. In the investigation, we found the self-similarity of the packet round trip time over IP which is our main result, and then we applied the self-similarity to the dynamic retransmission timeout design for TCP. We also found that our modeling is suited to the design of a packet delay buffer with a given reliability factor in a real-time communication application on the Internet.

Modern communication networks are based on the merging of computer communication technologies. As such, they use a wide variety of technologies, and are therefore subject to a number of failures. The users of such networks normally do not know, or care, whether network unavailability results from hardware malfunctions, software failures, protocol deadlock, excessive network congestion, error recovery delays, or other phenomena.

Some standard definitions of reliability, such as those based on the probability that all components of a system are operational at a given time, are not relevant to large telecommunication networks. In fact, many telecommunication networks are so large that the probability they are

operational according to standard criterion may be very near zero. Since telecommunication systems are repairable systems, they fall into a class of systems for which reliability theory techniques are surprisingly incomplete.

A new reliability technique is required for telecommunication networks. According to J. D. Spragins, J. C. Sinclair, Y. J. Kang, and H. Jafari [26], some important areas in need of research are:

(1) developing more efficient computational algorithms;

(2) exploiting network routing algorithms and similar protocols to help develop more realistic, and possibly simpler, reliability models;

(3) modeling the effects of statistical dependencies among failures of different network components;

(4) developing models that reflect the impact of several factors which are difficult to quantify but have major impact on network reliability;

(5) developing better models for software reliability and measuring their effect on overall network reliability;

(6) adequately modeling nodal reliabilities including effects of both hardware and software failures;

(7) finding techniques for including the impact of protocol related factors such as deadlock, routing, flow control, congestion and error recovery delays on network performance and perceived reliabilities and availabilities; and

(8) developing unified reliability and performance models.

It is noted that a large network (for example, the Internet) can be viewed as a black box. So one of major measurements for the performance of the network transmission reliability is constraint delay. For example, V. Jacobson suggests the idea of TCP's self-clock behavior to develop a more reliable transmission protocol [12]. In Chapter 4, we define the timing reliability of the system, which is modeled by a "place" stochastic Petri net, and provide models for finding the tightest constraint timing bounds and the tightest system constraint timing bound, respectively. To illustrate our modeling, we find the tightest deterministic timeout of TCP within an actual network. We discuss the real-time transport protocol briefly and describe that our modeling technique is suited to model real-time communication in the Internet. For the latter, the minimum size delay buffer is computed given a reliability factor.

In Chapter 5, we present the statistical data of the round trip time over IP in the Internet. We discovered that the round trip time of packets in the Internet displays the self-similar behavior. To our knowledge, no one has found and considered this property about the round-trip time of transmitted packets over IP. By intuitive reasoning and experimentation, we found that such a self-similarity could be used to estimate the parameters in the Jacobson's retransmission timeout algorithm. It follows that this property is applicable to other protocol design issues.

Most transmission timeouts in the Internet are the results of congestion. All the Internet TCP algorithms assume that timeouts are caused by congestion and monitor timeouts as a sign of trouble. Timeouts are directly related to transmission reliability and performance efficiency of the network. In October 1986, the Internet experienced the first of what became a series of "congestion collapses". During this period, the throughput from LBL to UC Berkeley (sites separated by

400 yards and three IMP hops) dropped from 32 Kbps to 40 bps. In response, Van Jacobson devised a new algorithm "Jacobson's algorithm" to adjust the performance of the network [12]. Variants of this algorithm are widely used by today's TCP implementations. The analysis of statistical distribution of the round trip time of packets in a proper large network provided another motivation to investigate TCP/IP in Chapter 5.

In Chapter 6, we present other observations on the round trip times of the transmitted packets over IP, such as the relationship between the loss rate and the Hurst parameter, the discussion of estimated number of nodes in the one way trip of a packet between two hosts, and monitored the packet loss rate over IP in different time periods.

In Chapter 7, we discuss the importance of our research and propose future research. Most importantly we found that the distribution of the round-trip time of the transmitted packets over IP in the Internet has the self-similar property, and illustrated that self-similarity could be applicable to protocol design. The figures we referred to in the thesis are numbered within each chapter.

Chapter 2

Concepts of Petri Nets

Petri nets, introduced by Carl Adam Petri in 1962 [20], are a graphic and mathematical modeling tool applicable to many systems. They are especially good for describing and studying information processing systems that are characterized as being concurrent, asynchronous, distributed, parallel, non-deterministic, and stochastic. As a graphical tool, Petri nets can be used as a visual communication aid similar to flow charts, block diagrams, and networks. In addition, tokens are used in these nets to simulate the dynamic and concurrent activities of systems. As a mathematical tool, it can be used to set up state equations, algebraic equations, and other mathematical models governing the behaviour of systems. Background information on the development of Petri net up to 1989 can be found in T. Murata [19].

2.1 Ordinary Petri Nets

2.1.1 Transition Enabling and Firing

In this section, we introduce the more important rule from Petri net theory: *the rule of transition enabling and firing*. A Petri net is a particular kind of directed graph, together with an initial state called the initial marking M_0. The underlying graph N of a Petri net is a directed, weighted, bipartite graph consisting of two types of nodes, called places and transitions, where arcs are either from a place to a transition or from a transition to a place. Arcs are labelled with their weights (positive integers). A marking (state) assigns to each place a non-negative integer. If a marking assigns to place p a non-negative integer k, we say that p *is marked with* k *tokens*. Pic-

torially, we place k dots (tokens) in place p. In modeling, using the concept of conditions and events, places represent conditions, and transitions represent events. A transition has a certain number of input and output places representing the pre-conditions and post-conditions of an event, respectively. The presence of a token in a place is interpreted as holding the truth of condition associated with the place. In another interpretation, k tokens are put in a place to indicate that k data items or resources are available.

A Petri net can be formally defined in the following manner:

Definition 1. A Petri net is a 5-tuple, $PN = (P, T, F, W, M_0)$, where

(1) $P = \{p_1, p_2, ..., p_m\}$ is a finite set of places;

(2) $T = \{t_1, t_2, ..., t_n\}$ is a finite set of transitions;

(3) $F \subseteq P \times T \cup T \times P$ is a set of arcs (flow relations);

(4) $W: F \rightarrow \{1, 2, 3, \ldots\}$ is a weight function;

(5) $M_0: P \rightarrow \{0, 1, 2, 3, \ldots\}$ is the initial marking;

(6) $P \cap T = \varnothing$ and $P \cup T \neq \varnothing$.

A Petri net structure $N = (P, T, F, W)$ without any specific initial marking is denoted by N. A Petri net with the given initial marking is denoted by (N, M_0).

The behaviour of many systems can be described in terms of system states and their changes. In order to simulate the dynamic behaviour of a system, a state or marking in a Petri net is changed according to the following transition firing rule:

Definition 2. Firing Rule:

(1) A transition t is said to be *enabled* if each input p of t is marked with at least $w(p, t)$ tokens, where $w(p, t)$ is the weight of the arc from p to t;

(2) An enabled transition may or may not fire depending on whether or not the event actually takes place;

(3) A firing of an enabled transition t removes $w(p, t)$ tokens from each input place p of t, where $w(p, t)$ is the weight of the arc from p to t and adds $w(t, p)$ tokens to each output place p from t, where $w(t, p)$ is the weight of the arc from t to p;

(4) A transition without any input place is called a *source transition*, and one without any output place is called a *sink transition*. Note that a source transition is unconditionally enabled, and that the firing of a sink transition consumes tokens but does not produce any.

2.1.2 Behavioural Properties

After modeling systems with Petri nets, one obvious question is "What can we do with the models?" A major strength of Petri nets is their support for analysis of many properties and problems associated with concurrent systems. Two types of properties can be studied with a Petri net model: those which depend on the initial marking, and those which are independent of the initial marking. The former type is referred to as marking-dependent or behavioral properties, whereas the latter type is called the structure property. In this subsection, we discuss some of the basic behavioral properties and their analysis problems.

2.1.2.1 Reachability

Reachability is a fundamental basis for studying the dynamic properties of any system.

The firing of an enabled transition will change the token distribution (marking) in a net according to its transition rules. A sequence of firing will result in a sequence of marking. A marking M_n is said to be *reachable from a marking* M_0 if there exists a sequence of firings that transforms M_0 to M_n. A firing sequence is denoted by $s = M_1\ t_1\ M_2\ t_2\ M_3\ \ldots\ t_n\ M_n$ or simply $s = t_1 t_2\ \ldots\ t_n$. In this case, M_n is reachable from M_0 by s. The set of all possible markings reachable from M_0 in a net (N, M_0) is denoted by $R(N, M_0)$ or simply $R(M_0)$. The set of all passible firing sequences from M_0 in a net (N, M_0) is denoted by $L(N, M_0)$ or simply $L(M_0)$. The reachability problem for Petri nets is the problem of finding if $M_n \in R(M_0)$ in a net (N, M_0) for a given marking M_n in a net (N, M_0).

2.1.2.2 Boundedness

A Petri net (N, M_0) is said to be *k-bounded* or *simply bounded* if the number of tokens in each place does not exceed a finite number k for any marking reachable from M_0. A Petri net (N, M_0) is said to be *safe* if it is 1-bounded.

2.1.2.3 Liveness

The concept of liveness is closely related to the absence of deadlocks. A Petri net (N, M_0) is said to be *live* (equivalently M_0 is said to be live for N) if, no matter what marking has been reached from M_0, it is possible to fire any transition of the net by progressing through some further firing sequences. This means that a live Petri net guarantees deadlock-free operation, no mat-

ter what firing sequence is chosen. Further details about other properties such as reversibility, coverability, persistence, and fairness can be found in Murata [19].

2.1.3 Methods of Analysis

Methods of analysis for Petri nets may be classified into three groups: the coverability (reachability) tree method, the matrix-equation approach, and the reduction or decomposition technique. The first method essentially involves the enumeration of all reachable markings or their coverable markings. It can be applied to all classes of nets, but it is limited to small nets due to the complexity of the state-space explosion. On the other hand, matrix equations and reduction techniques are powerful and are mainly applicable to special subclasses of Petri nets or special situations.

2.1.3.1 Coverability Tree

Given a Petri net (N, M_0), from the initial marking M_0, we can obtain as many new markings as the number of enabled transitions. From each new marking, we can again reach more markings. This process results in a tree representation of markings. Nodes represent markings generated from M_0 (the root) and its successors, and each arc represents a transition firing which transforms one marking to another. The coverability tree for a Petri net (N, M_0) can be constructed using the algorithm in Murata [19].

Some of the properties that can be studied by using the coverability tree for a Petri net (N, M_0) are the following [19]:

Theorem 1. A net (N, M_0) is safe if and only if 0's and 1's appear in the node labels in the coverability tree.

Theorem 2. A transition is dead if and only if it does not appear as an arc label in the coverability tree.

Theorem 3. If M is reachable from M_0, then there exists a node labelled M' such that $M \leq M'$.

2.1.3.2 Incidence Matrix and State Equations

In this subsection we present matrix equations that govern the dynamic behavior of concurrent systems modelled by Petri nets. For a Petri net N with n transitions and m places, the incidence matrix $A = [a_{ji}]$ is an $m \times n$ matrix of integers and its a_{ji} entry is given by

$$a_{ji} = a^{+}_{ji} - a^{-}_{ji}, \tag{1}$$

where $a^{+}_{ji} = w(i, j)$ is the weight of the arc from transition i to its output place j and $a^{-}_{ji} = w(j, i)$ is the weight of the arc from transition i to its input place j. It is easy to see that a^{-}_{ji}, a^{+}_{ji}, and a_{ji},

respectively, represent the number of tokens removed, added, and changed in place j when transition i fires once. Transition i is enabled at a marking M if and only if $a^{-}_{ji} \leq M(j)$ for all $j = 1, 2, \ldots, m$. In writing matrix equations, we write a marking M_k as an $m \times 1$ column vector. The j^{th} entry of M_k denotes the number of tokens in place j after the k^{th} firing in some firing sequence. The k^{th} firing or control vector $s_k = [0, \ldots, 0, 1, 0, \ldots, 0]$ is an $1 \times n$ vector with a 1 in the i^{th} position (indicating that transition i fires at the k^{th} firing) and with 0's in all other positions. Since the i^{th} column of the incidence matrix A denotes the change of the marking as the result of firing transition i, we can write the state equation for a Petri net as:

$$M_k = M_{k-1} + A \times s_k^{T}, k = 1, 2, \ldots \quad (2)$$

where s_k^{T} is the transpose of the row vector s_k. The necessary reachability conditions are given in [19].

2.1.3.3 Simple Reduction Rule for Analysis

This method facilitates the analysis of a large system by reducing the system model to a simpler one, while preserving the system properties to be analysed. There are many transformation techniques for Petri nets which can be used for analysing liveness, safeness, and boundedness. Figure 2.1 shows some examples for such transformations [19]

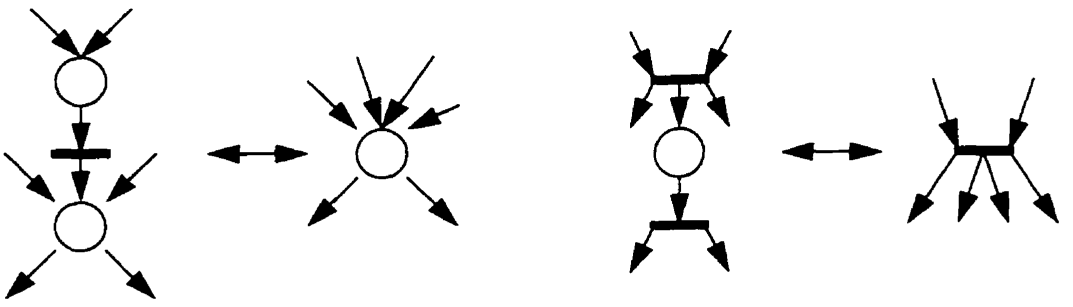


Figure 2.1 Some transformations Preserving Liveness, Safeness, and Boundedness

2.2 Stochastic Petri Net

Traditionally, a *stochastic Petri net* (SPN) is a 5-tuple $SPN = \langle P, T, F, M, G \rangle$, where $\langle P, T, F, M \rangle$ is a Petri net in a general sense, G is a function which maps a transition t to an exponentially distributed random variable that expresses the delay from the enabling to the firing of t. In a case where several transitions are simultaneously enabled, the transition that has the shortest delay will fire first. Due to the memoryless property of exponential distribution of firing delays, it has been shown that the reachability graph of a bounded SPN is isomorphic to a finite Markov Chain [17]. However, it can not be used to model a more realistic system with the random timing constraints on the places rather than transitions. In Chapter 4, we consider another kind of stochastic Petri net, introduced in [6], to model the system with the random timing constraints on the places, such as in a communication system.

Chapter 3

Overview of Transport Protocol: TCP

To achieve good performance for end systems and for connecting networks as a whole, the design and implementation of the transport protocol are vital ingredients. The transport protocol provides an interface between applications and the networking facility that enables the applications to request a desired quality of service. Connection-oriented transport protocols, such as TCP, divide the total flow of application data into disjoint logical streams and may allocate resources differentially among those streams. Finally, the transport protocol's policies for transmission and retransmission of data units have a profound impact on the level of congestion in the networking facility.

This chapter briefly examines an important transport protocol: TCP--transmission control protocol. TCP is the most widely used transport protocol, employed by the majority of applications that use the TCP/IP protocol suite. First, we introduce the concepts of TCP/IP. Then we outline the TCP flow and congestion control.

3.1 TCP/IP Protocol Suite

Two protocol architectures have served as the basis for the development of interoperable communications standards: the TCP/IP protocol suite and the OSI reference model. TCP/IP is the most widely used interoperable architecture, and the OSI has become the standard model for classifying communications functions.

TCP/IP is a result of protocol research and development conducted on the experimental packet-switched network, ARPANET, funded by the Defense Advanced Research Projects Agency (DARPA), and is generally referred to as the TCP/IP protocol suite. Based on the protocol standards that have been developed, we can organize the communication task for TCP/IP into five relatively independent layers:

- **Application layer:** provides communication between processes or applications on separate hosts.

- **Host-to-host, or transport layer:** provides end-to-end, data-transfer service. This layer may include reliability mechanisms. It hides the details of the underlying network or networks from the application layer.

- **Internet layer:** concerned with routing data from source to destination host through one or more networks connected by routers.

- **Network access layer:** concerned with logical interface between an end system and a subnetwork.

- **Physical layer:** defines characteristics of the transmission medium, signalling rate, and signal encoding scheme.

Figure 3.1 shows that the TCP/IP protocols are implemented in end systems. The physical and network access layers provide interaction between the end system and the network, whereas the transport and application layers are what is known as *end-to-end* protocols; they support inter-

action between two end systems. The internet layer has the flavour of both. At this layer, the end system communicates routing information to the network but also must provide some common functions between the two end systems.

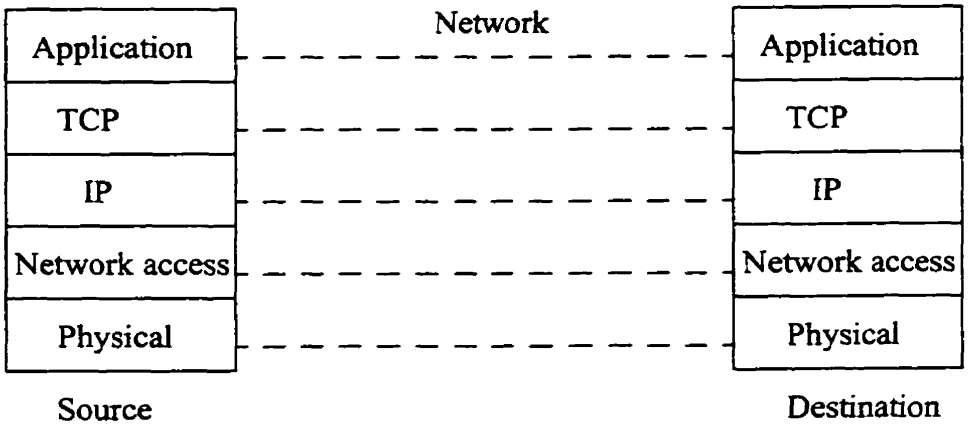


Figure 3.1 TCP/IP implementation in source and destination hosts

To control the operation of TCP and IP, control information as well as user data must be transmitted, as suggested in Figure 3.2. The sending process generates a block of data and passes this to TCP. TCP may break this block into smaller pieces to make it more manageable. To each of these pieces, TCP appends control information known as the TCP header, thereby forming a TCP segment. The control information is to be used by the peer TCP protocol entity at the receiving side host.

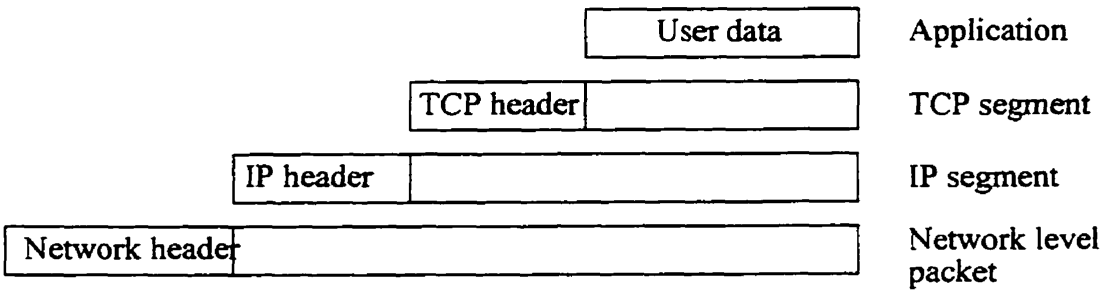


Figure 3.2 Protocol data units in the TCP/IP architecture

3.2 Transmission Control Protocol (TCP)

The current version of TCP is officially defined in RFC 793. A number of enhancements and implementation specifications were subsequently added. Those that are required for a conformant implementation of TCP as of 1989 are documented in RFC 1122. Since then, a number of additional changes have been documented. TCP is a complex protocol. It was designed to accomplish three major objectives:

- **In-order delivery:** deliver data to the receiving application in the same sequence as transmitted by the sending application.

- **Byte-stream model:** allows the sender and receiver view the data simply as a series of bytes without apparent boundary points.

- **Reliable data delivery:** ensure all of the data transmitted arrives at the receiver with its original contents.

In the following, we outline TCP flow and congestion control mechanisms.

3.2.1 TCP Flow Control

TCP uses a form of sliding-window mechanism to provide flow control as in a data link control protocol. This mechanism is known as a *credit* allocation scheme. It allows the sender to send as many packets as the receiver can accommodate. For this scheme, each individual byte of data that is transmitted is considered to have a sequence number. When a TCP entity sends a segment, it includes the sequence number of the first byte in the segment data field. A TCP entity acknowledges an incoming segment with a message of the form ($A = i$, $W = j$), with the following interpretation:

- All bytes through sequence number $i - 1$ are acknowledged; the next expected byte has sequence number i.
- Permission is granted to send an additional window W (called *offered window*) of j bytes of data; that is, the j bytes corresponding to sequence numbers i through $i + j - 1$.

In the case of TCP, there is no explicit negative acknowledgment, such as the REJ or SREJ found in link control protocols, where REJ is in a negative ACK packet for rejecting and SREJ means selective REJ. TCP relies exclusively on positive acknowledgment and retransmission when an acknowledgment does not arrive within a given timeout period.

All TCP implementations attempt to estimate the current round-trip delay by observing the pattern of delay for recent segments, and then the timer is set to a value somewhat greater than the estimated round-trip delay. Details will be discussed in Section 5.3.

3.2.2 TCP Congestion Control

Congestion control is the vital performance issue in the Internet. The limitation on how fast the sender should transmit may be derived from both the limited buffer at the receiver as well as the limited capacity inside the network. Originally, TCP provided congestion control by setting the retransmission timeout (RTO) to a multiple of the estimated mean round-trip time (RTT). When the RTO expired, unacknowledged packets were retransmitted, and the RTO was doubled. During periods of high congestion, the connection would progressively lower its sending rate.

In a historic paper [12], Jacobson described the shortcomings of this form of congestion control: in particular, its excessive consumption of resources due to retransmitting multiple packets, and the instability that occurs because it does so precisely when the network has been overloaded to the point of packet loss. He also identified inadequacies in the RTO algorithm, which used only the estimated mean RTT, without including an estimated RTT variance. He addressed these problems by introducing a second window, the *congestion window* (*cwnd*), and a modified RTO algorithm that includes the estimated RTT variance. Without them, the network would inevitably devolve into "congestion collapse".

The *cwnd* is completely separate from the receiver's offered window. The *offered window* governs how much "in-flight" data the receiver's buffer can accommodate, and the *cwnd* governs how much the buffers along the network path can accommodate. Jacobson discussed two different issues in managing the *cwnd*. The first is what value to use for it initially. The second is how it should be cut to adapt to congestion upon detecting loss.

A solution for the first issue is the *slow start* mechanism which probes the Internet to make sure that it is not sending too many segments into an already congested environment. A solution for the second issue is the *dynamic window sizing* technique on congestion, which controls *cwnd* growing by dynamically setting the TCP state variable *ssthresh* (slow start threshold) and corporating with RTT and RTO.

In addition to *slow start*, *dynamic window sizing*, and retransmission timer management, the current TCP uses Karn's algorithm, fast retransmit algorithm, and fast recovery algorithm. The details can be found in Stevens's book [28].

Chapter 4

A Probabilistic Timing Analysis for Synthesis in Reliable Transmission Design

In this chapter, we model a system or network with the timing constraints on the states. For the system with random variable constraints, the timing analysis for synthesis finds the tightest bounds on those variables which satisfy the timing constraints given in the specifications. We model such systems with "Place Stochastic Petri Nets" which allows the designer to perform a reliability analysis in addition to finding bounds for timing constraints. As an illustration, we present an analysis of the TCP with a deterministic timeout design and delay buffer design in real time communication in the Internet.

Our goal in the current chapter is network reliability modeling, in particular reliable transmission design. As mentioned, the initial idea came from the timing analysis for synthesis in microprocessor interface design which was introduced by Marco A. Escalante and Nikitas J. Dimopoulos [6]. To illustrate our use of the technique, we investigated the timeout design in TCP and a delay buffer design in a real-time communication in the Internet. In this chapter, we only present how to find the tightest timeout bound for the deterministic retransmission timeout design as an illustration of our modification and application of the technique. For a real-time communication, we discuss that our modeling is suited to design of a delay buffer with a given reliability factor and compute the minimum size of delay buffer.

Section 4.1 reviews the stochastic Petri net model and the computation of delay between

transitions. Section 4.2 defines the timing reliability under given reliability factors on the timing constraints and system tightest timing constraint. It also discusses how to simplify the TCP Petri net model with our technique. In section 4.3, the data on round trip time measurement over IP is given to show the method of finding tightest timeout. In section 4, we describe how the delay buffer for real time communication in the Internet can be modelled and designed.

4.1 Timing Analysis

In this section, we list some basic notations, definitions, and a short description of computing the delay between two transitions, by examples. The details can be found in [6, 7, 8].

4.1.1 Traditional Stochastic Petri Net

As stated in Chapter 2, a traditional stochastic Petri net (SPN) is a 5-tuple $SPN = \langle P, T, F, M, G \rangle$, where $\langle P, T, F, M \rangle$ is a Petri net in general sense, G is a function which maps a transition t to an exponentially distributed random variable that expresses the delay from the enabling to the firing of t. In a case where several transitions are simultaneously enabled, the transition that has the shortest delay will fire first (ref. Figure 1(a)). Due to the memoryless property of exponential distribution of firing delays, it has been shown that the reachability graph of a bounded SPN is isomorphic to a finite Markov Chain. However, we need a model with the random timing constraints on the places other than transitions.

4.1.2 Place Stochastic Petri Net:

A place stochastic Petri net, introduced by Marco A. Escalante and Nikitas J. Dimopoulos

in [6], models the system using the random timing constraints on the places, such as in a communication network. We would like to use the term Stochastic Petri net for Place Stochastic petri net in the following context.

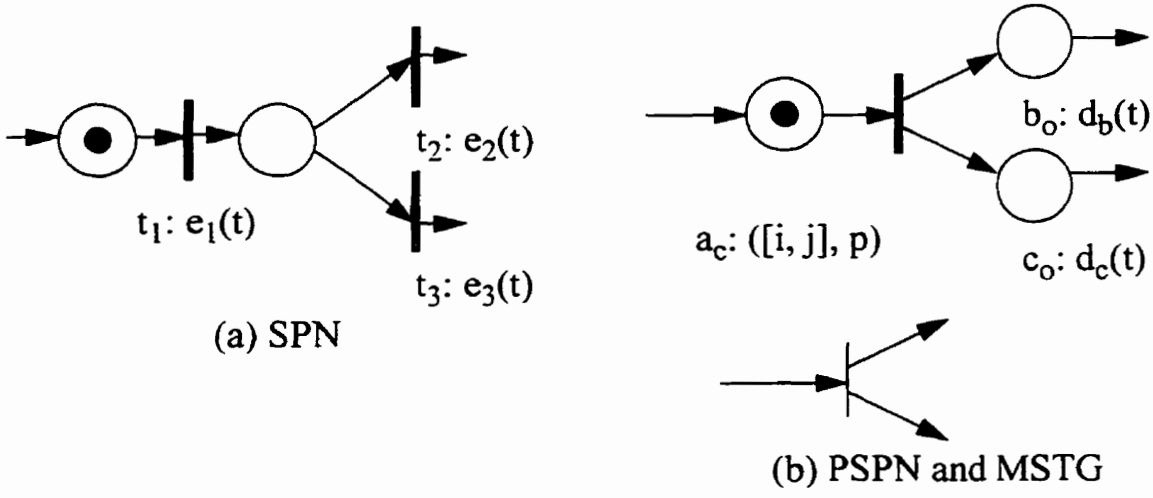


Figure 4.1 Stochastic Petri Net

Definition 1. A *stochastic Petri net model* is a 6-tuple $PN = \langle P, T, F, M, G, \Delta \rangle$ where P is a non-empty set of places partitioned into two sets P_o (*operational*) and P_c (*constraint*); T is a non-empty set of transitions; $F \subseteq (P \times T) \cup (T \times P)$ is the flow relation; M is the marking function from P to the set N of the natural numbers; G is the operational labelling function from P_c to V, which assigns to each operational place $p_i \in P_o$ a random variable t_i with probability density function $v_i = f_i(t_i)$; and Δ is the constraint labelling function from P_c to $I \times [0,1]$, which assigns to each constraint place $p_i \in P_c$ a closed interval $I_i \in I$ and a reliability figure r_i (I is the set of closed

intervals and V is the set of pairs of random variables t_i and probability density functions $v_i = f_i(t_i)$). The *preset (postset)* of a transition t is the set of incoming places to (outgoing places from) t and is denoted $\bullet t$ ($t \bullet$) (ref. Figure 4.1(b)).

Definition 2. Firing Rule: (1) A transition t is enabled when every incoming place p of t contains a token. (2) An enabled transition t fires immediately. When it fires, the transition sends tokens to every outgoing place p of t and anti-tokens to every incoming place p of t. (3) An operation place p labelled with the random variable t_p and the corresponding distribution $f_p(t_p)$, upon receiving a token at time t, makes it visible to all outgoing transitions of p at time $t + t_p$. The token is held by the place until it is annihilated by an anti-token. (4) A constraint place p labelled $\Delta_p = [a, b]$, upon receiving a token at time t, holds it during the interval $[t + a, t + b]$.

Note that the firing rule implies that there is no selection for the enabled transition and the enabled transition must fire without any condition, whereas the traditional firing rule says that the enabled transition may or may not fire at will. The current firing rule therefore eliminates some minor transitions.

For efficient calculation, we introduce the concept of the marked Petri net (i.e., marked graph). The first half of the following is a standard definition in the literature of Petri nets.

Definition 3 . A *marked Petri net* is a Petri net such that each place has exactly one input transition and output transition. A *marked stochastic transition graph* (MSTG) is the *marked stochastic* Petri net with the associated graph in which a link represents a place (ref. Figure 4.1(b)).

The MSTG is a subclass of STG, and makes the transitions flow more smoothly. For a stochastic Petri net in a complicated system, we can consider several MSTG subnets, and then integrate the whole net.

4.1.3 Computation of the Delay Between Transitions:

This subsection introduces a procedure for determining the delay between two transitions of a MSTG in [6]. Places can be drawn as links between transitions. Labels associated with constraint places and operational places are denoted by Δ and t, respectively. In Figure 4.2, places are associated with random variables t_i and the corresponding distributions $v_i = f_i(t_i)$. After the firing of a transition, say a at time t_a, a token is made visible to transition d at time $t_a + t_1$, where the pdf of t_1 is v_1. As the firing rule,

$$t_d = max(t_a + t_1, t_b + t_2, t_c + t_3). \tag{3}$$

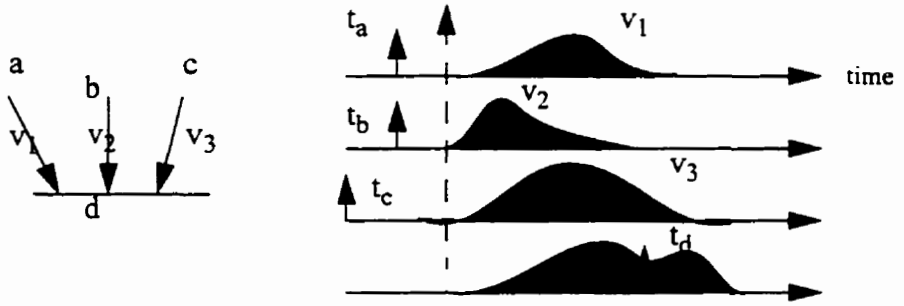


Figure 4.2 Modeling delays between transitions

Figure 4.3 shows a protocol and its corresponding unfolded graph [8]. The time of occurrence of any event is computed starting at time zero in topological order and assigning a time interval of occurrence to each transition in the graph, denoted by v_i (not to be confused with the above notation). Because the behaviour is periodic, the time separation between two events can be computed relative to the corresponding fork transition. The separation between transitions $b+^i$ and $a+^i$ for any cycle i ($i > 0$) is within $max\,(v_2 + v_4 + v_1, v_3 + v_5) - (v_2 + v_5)$, where the operations are on the intervals. Their fork transition is $b+^{i-1}$.

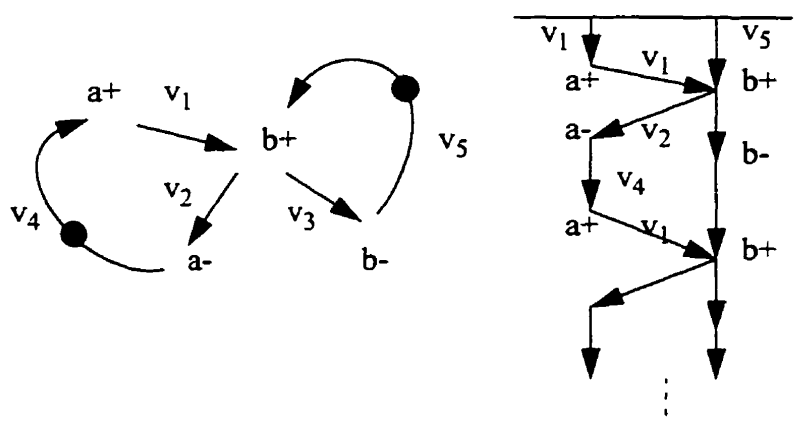


Figure 4.3 A signal transition graph of protocol and its unfolded graph

4.2. Timing Analysis for Synthesis:

This section first defines "timing reliability" and then determines the tightest timeout

bound for the deterministic retransmission timeout design of TCP at the end host. The more general transmission control protocol with more than one constraint places is included in our model. Finally, we give a formula to decide the shortest timing constraint of a system under given reliability factors.

Definition 4. In a MSTG for a given system, $\Delta_i = [a_i, b_i]$ and r_i $(i > 0)$ are the constraint labels, where r_i are reliability factors. Assume that for the constraint place from transition A_i to B_i with Δ_i and r_i, we have the probability density function $f_i(z)$ of the time separation $z = t_A - t_B$ between A_i and B_i. The stochastic STG is called *timing-reliable under these reliability factors* r_i if $\int_{\Delta_i} f_i(z)dz \geq r_i$ is true for all i.

From these reliability factors, it is possible to get the overall reliability based on classic reliability theory.

An example of the timeout constraint of TCP will be considered. First the Petri net model of alternating bit protocol with unnumbered ACKs as shown in Figure 4.4 is simplified by reducing two cycles to one and eliminating the minor transition branches such that we can apply MSTG (Figure 4.5). The MSTG can be drawn as shown in Figure 4.6. From the distributions v_2, v_3 and v_4 we can get the distribution $f(z)$ for $z = t_{A+} - t_{A-}$. Let $\Delta = [a, b]$ with the reliability factor r. Then we can find the tightest timeout bound min(b) under reliability factor r by:

min (b) subject to

$$\int_{\Delta} f(z)dz \geq r \ . \tag{4}$$

The Petri net in Figure 4.4 is not a marked Petri net, but Figure 4.5 is. Figure 4.5 models not only alternating bit protocol but also other TCP protocols. Sometimes, the type of the distribution $f(z, x)$ of z is known with an unknown parameter vector x. For this case, one needs to find a suitable x by optimization and then look for min b.

For extended TCP including TCP in mobile communication, we have the following. Let $\Delta_0 = [a_0, b_0]$ and r be the constraint labels for the place of waiting ACK. Let $\Delta_i = [a_i, b_i]$ and r_i $(i > 0)$ be the other constraint labels. Thus, the timing analysis for synthesis is the optimization problem (for fixed $a_0 \geq 0$):

$$\min \ b_0 \text{ subject to}$$

$$\int_{\Delta_i} f_i(z)dz \geq r_i \text{ for all i.} \tag{5}$$

For a system, let $\Delta_i = [a_i, b_i]$ and r_i $(i > 0)$ be all constraint labels, where $a_i \geq 0$. Then the tightest system constraint bound is the optimization problem:

$$\min (b_1 + b_2 + \ldots + b_n) \text{ subject to } \int_{\Delta_i} f_i(z)dz \geq r_i \text{ for all } i. \tag{6}$$

By using the technique in subsection 4.1.3, the timeout bound can be tested to see whether or not it satisfies the reliability constraints by considering the constraint equations. The testing can be considered at any place in a cycle of transitions in the MSTG.

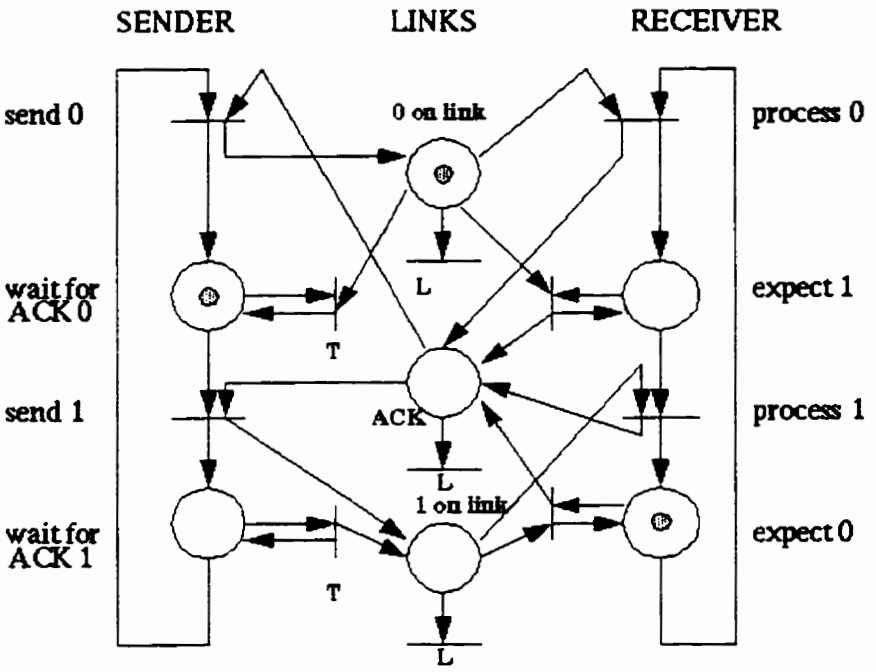


Figure 4.4 Petri net model of alternating bit protocol with unnumbered ACKs

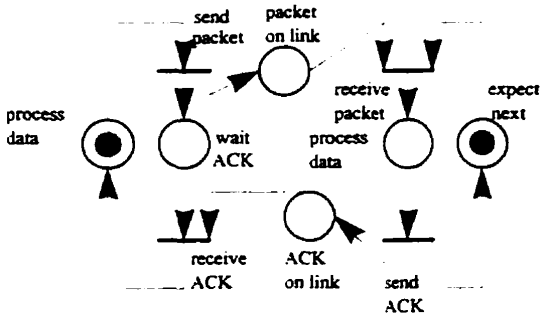


Figure 4.5 Simplified protocol

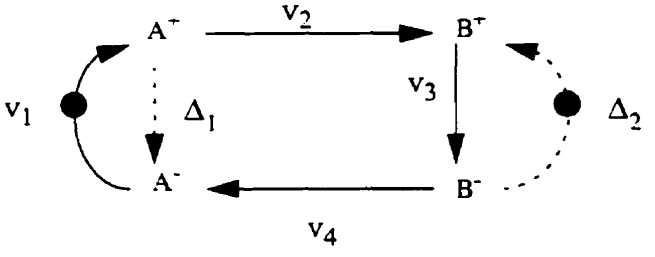


Figure 4.6 Corresponding stochastic STG

Note that the timeout transition is replaced by the constraint reliability factor in the above TCP example. Since exact measuring of a one way trip time in the Internet is impractical, the distribution of the round-trip time was measured. The following section presents the analysis of the measured round-trip time data in an actual wide area network.

4.3 Distribution of Round Trip Time and Deterministic Timeout Design

Designing the timeout in TCP requires collecting the round-trip time data over the unreli-

able layer IP. The ICMP and UDP can be used to implement this requirement over the Internet.

The data of round trip times was collected for fixed size packets from ic17 to www.nba.com with 14 hops. The data from ic17 to some intermediate nodes was also collected. The probability density distributions and the tables of the tightest upper bounds for given reliability values were established.

The one way topology of the measured network is shown in Figure 4.7. The duration of measurement was from 10:00am to 4:00pm. A packet with 100 byte data was transmitted once per second to the destination. There were 1000 packets transmitted with a small percentage lost on the way.

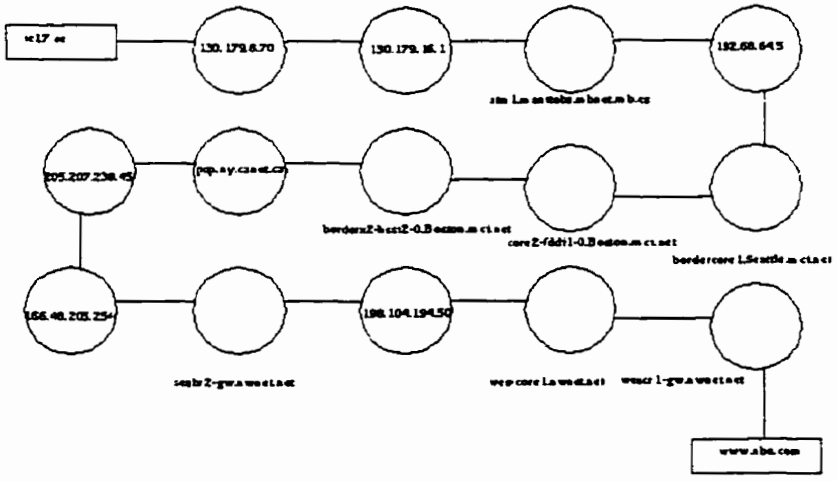


Figure 4.7 One way topology of the test-bed

Three sites were pinged: atm1.manitoba.mbnet.mb.ca; borderx2-hssi2-0.Boston.mci.net;

and www.nba.com. Their probability density distributions (PDD) curves are shown in clockwise order in Figure 4.8. From these curves and probability theory, it appears that the round-trip-time is subject to the distribution of an aggregated process dominated by Gamma distribution plus a heavy tail:

$$\left((\lambda t)^{k-1} \cdot \lambda e^{-\lambda t} \right) / (k-1)! \qquad (7)$$

where k is the number of intermediate nodes and these nodes are subject to the same exponential distribution $e^{\lambda t}$. Background material of formula (7) is discussed in Section 6.2.1.

Table 4.1 lists the tightest timeouts with the given reliability factors for www.nba.com.

Table 1:

Tightest bound(ms)	209	191	185	181	179	175	174	171	169	168
Reliability	0.99	0.98	0.97	0.96	0.95	0.94	0.93	0.92	0.91	0.90

166	165	165	165	163	160	160	159	158	158
0.89	0.88	0.87	0.86	0.85	0.84	0.83	0.82	0.81	0.80

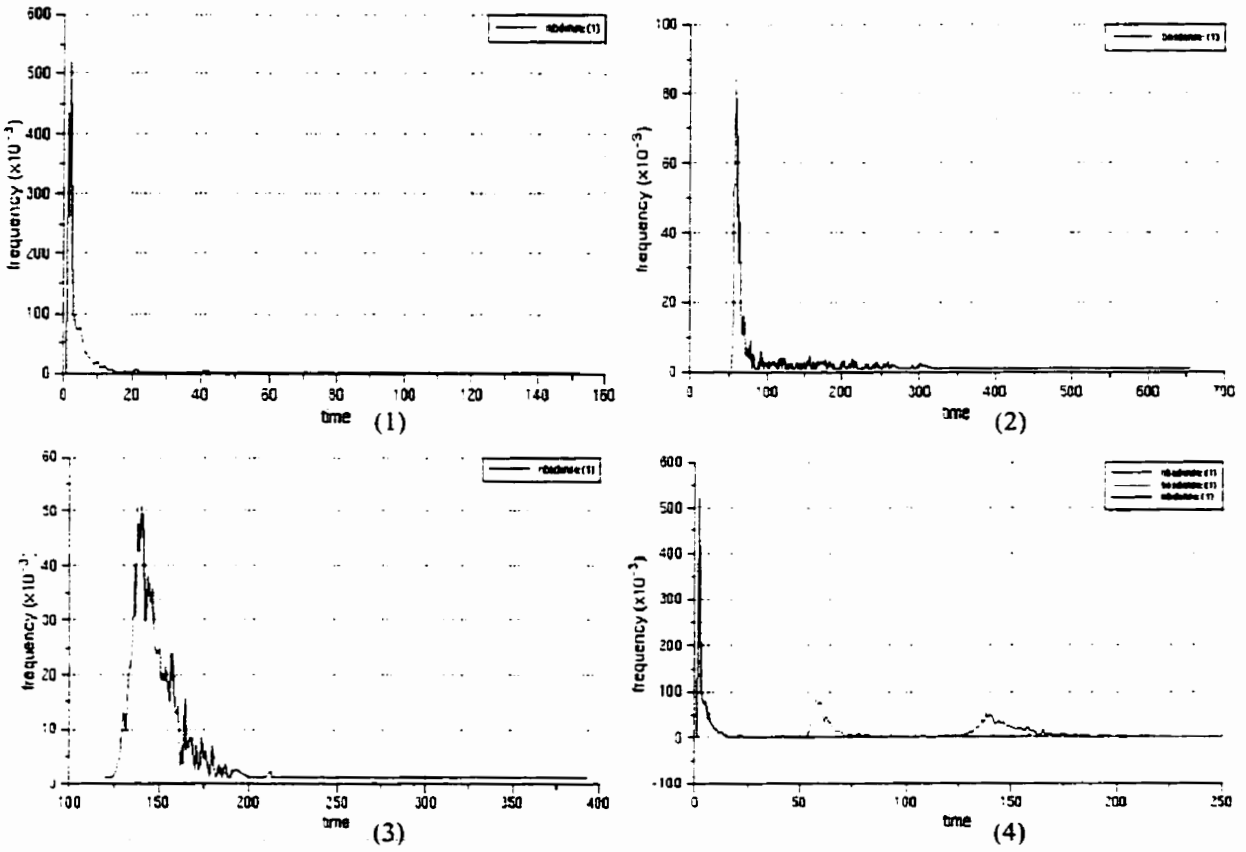


Figure 4.8 Distribution and comparison

4.3. Applications to Real-time Communication

A real-time distributed application is one in which a source is generating a stream of data at a constant rate and delivering that data to one or more destinations at the same constant rate. Examples of applications include audio and video conferencing. Although each real-time application could include its own mechanisms for supporting real-time transport, there are a number of common features that warrant the definition of a common protocol. A protocol designed for this purpose is the real-time transport protocol, defined in RFC 1889 [23]. This protocol consists mainly

of two protocols, RTP which is a data transfer protocol, and RTCP (RTP Control Protocol). Since real-time applications are more concerned with timing issues, our model is well suited for this situation.

An illustration of real-time traffic is shown in Figure 4.9. A constant flow of packets is generated at the source, each packet contains M bytes of data and one packet is sent out per T seconds. Because of variable delay and packet loss through the Internet, the inter-arrival times between packets are not maintained at the fixed T seconds at the destination. To solve this problem, the incoming packets need to be buffered, delayed slightly, and then released at the original constant rate to the application software.

Designing the time delay buffer so that the incoming packets have the same pacing time as the original is a problem to find tightest constraint bound under given reliability factors. Let $f(z)$ be the probability density function of the inter-arrival time between two transmitted packets from the source to the destination. If we require the satisfaction of the packets arrival being 90% in the real-time transmission requirement, then take the reliability $r = 0.90$ and the tightest delay of delay buffer is the minimization problem:

$$\min (b) \text{ subject to}$$

$$\int_{[0,\, b]} f(z)dz \geq r \tag{8}$$

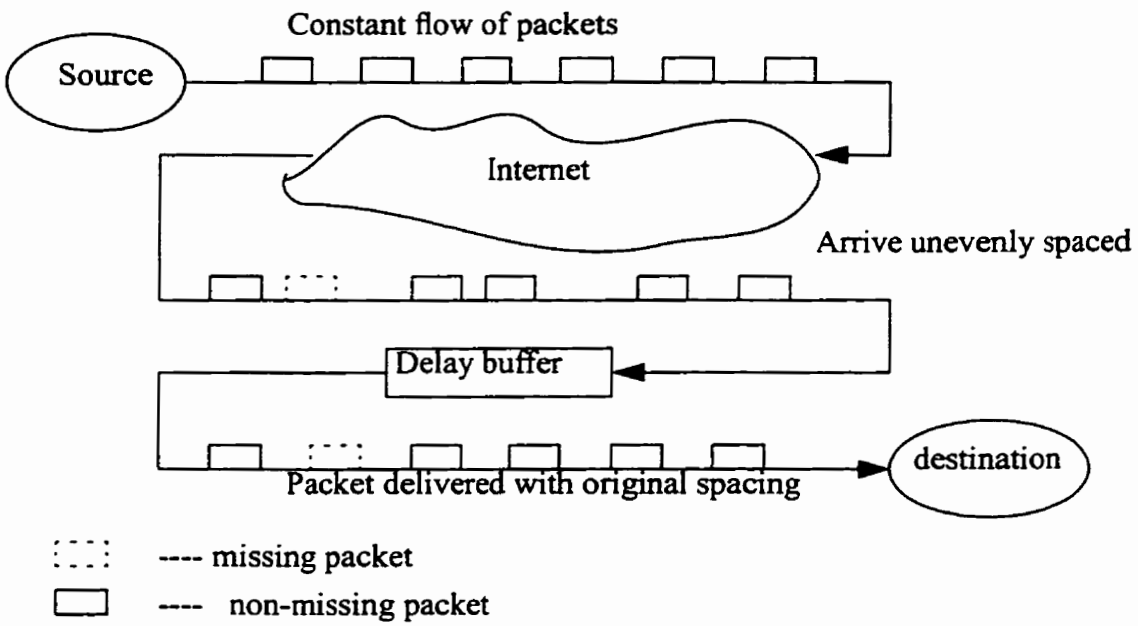


Figure 4.9 Real-time traffic

From the value of min (*b*), we can in turn decide at least how long the buffer is required to be to release a packet so that the incoming packets have the same pacing time as the original and design the smallest delay buffer size which improves the utilization of the communication system. The simple Petri net for this process is as shown in Figure 4.9.

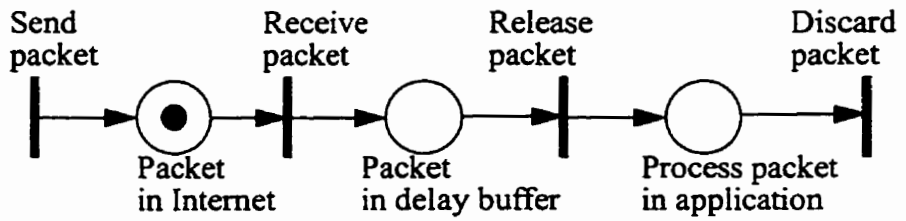


Figure 4.10 Simple Petri net model for real-time communication

For the above assumption of parameters *M* and *T*, the tightest delay of delay buffer is *D*=

max{min(b), T} and the smallest buffer size is $S = M\left\lceil \frac{D}{T} \right\rceil$ (bytes), where [ ] is the ceiling function. If the size of packet varies in the stream of data, we choose M as the maximum size of transmitted packet.

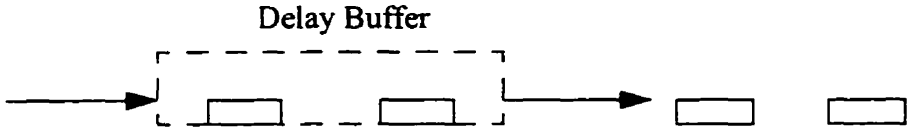


Figure 4.11 Delay buffer in the end system for real-time communication

4.5. Summary of Chapter 4

The modeling and analysis methods discussed in this chapter can be applied to the design of a deterministic reliable transmission protocol and the design of delay buffers for real-time communication, as demonstrated. The calculation for the design of delay buffer is a preliminary result. Further study is required to measure the inter-arrival time between packets and analyze the data. Since RTP is over UDP, we need to develop tools to catch packets or use measurement tools such as *snoop*. We believe that following the line described in this chapter, fruitful results related to timing constraints in real-time systems will be obtained.

Chapter 5

Self-Similarity of the Round Trip Time and the Jacobson's Retransmission Algorithm

This chapter covers two main topics which are based on previous research [30]. One is that the round trip times of IP packets are shown to be self-similar. The other is an application of the self-similarity to the timer algorithm of TCP.

Many recent studies of traffic measurements on a variety of packet networks have demonstrated that actual network traffic is self-similar in nature. However, effectively designing protocols that take self-similarity into account remains an open question. This chapter provides a method to apply self-similarity to the Jacobson's retransmission control algorithm of TCP. Firstly, the round trip time over IP in the Internet was found to be self-similar. Secondly, such a characteristic was used to estimate the parameters in the Jacobson's algorithm. This illustrates that the method is potentially applicable to other protocol designs that are related to self-similar delay.

5.1 Introduction

In the last four years, studies on packet networks have reported that actual network traffic is self-similar (these traffic studies include LAN, WAN, ATM, WWW [4, 16, 21]), in term of the distribution of packets/time unit vs. time. Although these findings in general can be expected to favor the use of self-similar models over traditional models, there has been considerable resistance to self-similar traffic modeling on practical grounds. According to W. Willinger, M. S.

Taqqu, and R. Sherman, and D. V. Wilson as indicated in [32], one of the major reasons for this resistance has been the absence of satisfactory answers to the following two questions: (1) What is the physical "explanation" for observed self-similar nature of measured traffic from today's packet networks? (2) What is the impact of self-similarity on network and protocol design and performance analysis? The answer to the first question, according to W. Willinger, M. S. Taqqu, and R. Sherman, and D. V. Wilson [32] was providing appropriate mathematical results and validating their findings with detailed statistical analyses of high time-resolution Ethernet traffic measurements. In an influential paper [35], Norros first attempted to develop analytic models of self-similar behavior. He extended the classic result about buffer requirement as a function of the mean utilization ρ in queuing theory with self-similarity consideration under certain conditions. The second part of this chapter is motivated by the second question above.

In this chapter, first we show that the round-trip time of transmitted packets in the Internet has the self-similar behavior. We are interested in utilizing the self-similar behavior to assist in determining design parameters useful within a higher level protocol. An initial attempt involved analyzing the round trip time between hosts in the Internet in terms of the Hurst parameter characterization. This round trip time was shown to exhibit self-similar behavior with a Hurst parameter between 0.60 and 0.93 after more than 400 tests. As a design parameter, the TCP retransmission timeout (RTO) was selected. The retransmission timeout is an interesting parameter to attempt to optimize in that it was proved to have a dramatic impact upon performance. If the RTO is set too long, the wait for packets to be retransmitted is excessive. On the other hand, if the RTO is set too small, packets are redundantly retransmitted, accumulatively causing heavier congestion. Although not optimal, the use of the Hurst characteristic was illustrated to modestly improve the

estimate of the RTO parameter of a protocol such as TCP. This result is encouraging because other design parameters can most likely be designed using information about the self-similar nature of traffic observations.

This chapter is organized into five sections. Section 5.2 reviews self-similarity and presents the self-similarity of round trip time over IP in the Internet in terms of the Hurst parameters. Section 5.3 applies the Hurst parameter of the round trip time to the Jacobson's retransmission timeout algorithm and proposes recommendation for finding the estimate values of parameters, based on our intuitive reasoning and experiments. Section 5.4 compares the performance of Jacobson's algorithm using different parameters, by way of examples. Section 5.5 summarizes our method and suggests how it is potentially applicable to other protocols.

5.2. The Self-Similarity Property:

A common definition of self-similarity for continuous-time stochastic processes is based on a direct scaling of the continuous time variable, as follows. A stochastic process $x(t)$ is statistically self-similar with parameter H ($0.5 \leq H < 1$) if for any real $a > 0$, the process $a^{-H} x(at)$ has the same statistical properties as $x(t)$. The parameter H, known as the Hurst parameter, or the self-similarity parameter, is a key measure of self-similarity. More precisely, H is a measure of the persistence of a statistical phenomenon and a measure of the length of the long-range dependence of a stochastic process. A value of $H = 0.5$ indicates the absence of self-similarity. The closer H is to 1, the greater the degree of persistence of long-range dependence. William Stallings summarizes these in [27].

Definitions of self-similarity

For a discrete-time stationary series x, x is said to be *exactly self-similar* with parameter $\beta\ (0 < \beta < 1)$, if for all $m = 1,2,...$ we have

$$Var(x^{(m)}) = (Var(x))/m^{\beta} \qquad Variance \tag{9}$$

$$R_{x^{(m)}}(k) = R_x(k) \qquad Autocorrelation \tag{10}$$

where $x^{(m)} = \{x_k^{(m)} \mid k = 0, 1, 2, \ldots\}$ is the m-aggregated time series by summing the original time series over non-overlapping, adjacent blocks of size m which is defined as follows:

$$x_k^{(m)} = \frac{1}{m} \sum_{i = km - (m-1)}^{km} x_i. \tag{11}$$

The parameter β is related to the Hurst parameter

$$H = 1 - \frac{\beta}{2}. \tag{12}$$

A process x is said to be *asymptotically self-similar* if for all m large enough

$$Var\left(x^{(m)}\right) = (Var(x))/m^{\beta} \qquad Variance \tag{13}$$

$$R_{x^{(m)}}(k) \to R_x(k) \qquad as \qquad m \to \infty \tag{14}$$

R/S Plot

There are a number of approaches to estimate the self-similarity parameter H. We will use an approach of R/S plot described as follows. Given a stochastic process $x(k)$ defined at discrete time instances $\{x_k, k=0, 1, 2, ...\}$ and any integer N, the ratio R/S is defined as:

$$\frac{R}{S} = \left(max\left\{ \sum_{k=1}^{j} \left(x_k - M(N) \right), 1 \le j \le N \right\} - min\left\{ \sum_{k=1}^{j} \left(x_k - M(N) \right), 1 \le j \le N \right\} \right) \Big/ \left(\sqrt{\frac{1}{N} \sum_{j=1}^{N} \left(x_k - M(N) \right)^2} \right) \quad (15)$$

where $M(N)$ is the sample mean over the time period N:

$$M(N) = \frac{1}{N} \sum_{j=1}^{N} x_j \quad (16)$$

The numerator in the ratio is a measure of the range of the process and the denominator is the sample standard deviation. For a self-similar process with $H > 0.5$, the ratio has the following characteristic for large N (W. Stallings [27]):

$$\frac{R}{S} \approx \left(\frac{N}{2} \right)^H \quad (17)$$

This can be rewritten as

$$\log\left[\frac{R}{S} \right] \approx H \log N - H \log 2 \quad (18)$$

If we plot R/S versus N on a *log-log* graph, the result should fit a straight line with slope H.

Self-similarity of round-trip time

Designing the timeout in TCP requires collecting the round-trip time data over the unreli-

able layer IP. The ICMP and UDP can be used to implement this requirement over the Internet. Based on a variety of statistical tests (by pinging the routers or server stations, more than 400 tests were performed) in the test-bed as shown in Figure 5.1, the distribution of the round-trip time was found to be self-similar with a Hurst parameter $0.60 \le H \le 0.93$, where H is computed by R/S plot. The Hurst parameter H changes slightly with the number of packets in the statistical data. It is bigger in a congestion situation (i.e., high loss rate). It also changes with the number of the routers between hosts. The results in Figure 5.2 represent the transmission of 100,000 packets, one per second, from the station ic13 in the Department of Electrical and Computer Engineering to the station charlotte in the University of California, Irvine. There were only 89037 responses and the Hurst parameter was H=0.866777. Figure 5.2 (b) is a segment of (a) and (c) is a segment of (b).

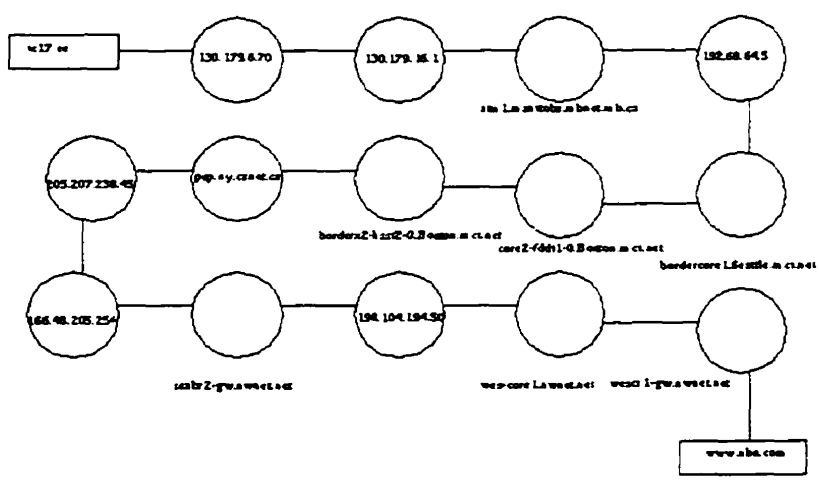


Figure 5.1 The test-bed for most experiments

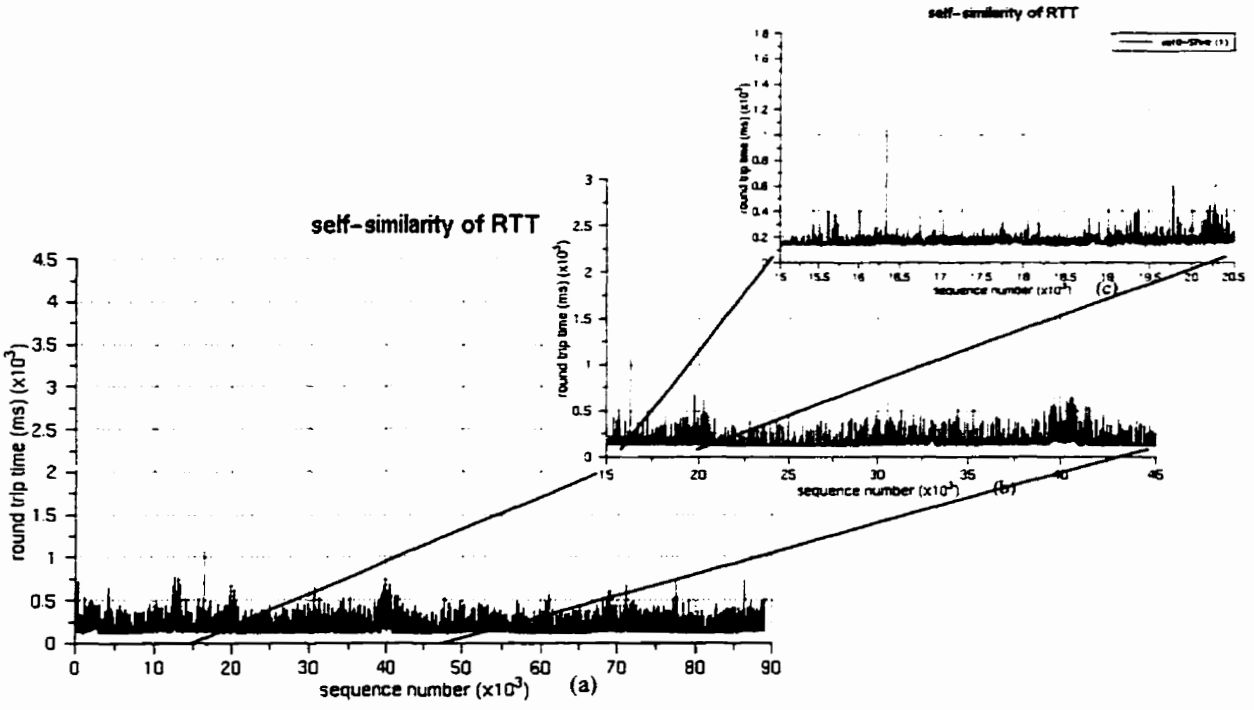


Figure 5.2 Self-Similarity of the Round Trip Time

5.3. Parameters of Jacobson Algorithm and the Hurst Parameter

It is well-known that deciding on the timeout interval is difficult for WANs. Most TCP implementations now use Jacobson's algorithm or one of its variations to dynamically set the timeout interval [12, 27]. The principle involves using new smoothing average round-trip time estimate *SRTTnew* and the new smoothing mean deviation estimate *SDEVnew* to predict the next retransmission timeout:

$$RTO = SRTTnew + f \times SDEVnew \tag{19}$$

The *SRTTnew* and *SDEVnew* are estimated from the old estimation of smoothing average round-trip time estimate *SRTTold*, the current round-trip time *RTT* and smoothing error *SERR* in the following:

$$SRTTnew = (1-g) \times SRTTold + g \times RTTcurrent \tag{20}$$

$$SDEVnew = (1-h) \times SDEVold + h \times |SERRcurrent| \tag{21}$$

$$SERRcurrent = RTTcurrent - SRTTold \tag{22}$$

where the parameters *g*, *h*, and *f* are determined by Jacobson's timing experiments as follows: $g = 1/8 = 0.125$, $h = 1/4 = 0.25$, and $f = 4$. One of the considerations for using 4 and 8 was the use of the shift operation in TCP implementations.

A common question remaining is whether there is a reasonable way to estimate the parameters based on the network characteristics. Intuitively, the Hurst parameter *H* is proportional to the degree of self-similarity which in turn is proportional to the amount of information contained in all consecutive subprocesses. Therefore, *H* is proportional to the weight of the previous segment because it is proportional to the amount of information in the previous segment of whole sequence. After lots of experiments, we found that the relative optimized values for estimating *g* and *h* in form of $\left(2^n - 1\right)/2^n$ are very near the Hurst parameter *H*. This consideration and experiment lead to the following recommendation for estimating the values *g* and *h* in (20), (21), and (22).

Recommendation for finding parameters g and h. Compute the Hurst parameter H. Set an integer N > 2. For the parameter H, there exists a positive integer n such that

$$\frac{2^n - 1}{2^n} < H \leq \frac{2^{n+1} - 1}{2^{n+1}} \quad and \quad \frac{1}{2^{n+1}} < \frac{1}{2^N} \tag{23}$$

Then take r as one of the two side values in the above inequality (the right side, usually, is slightly better). Let $g = 1 - r$ and $h = g/2$. For more universal use (e.g., the Internet), replace H in the above inequality by mean(H) + sdev(H). Usually, we suggest to choose N = 3 or 4 for the Internet, and n is first selected as the smallest value which satisfies (23) and then as the second smallest and so on. Select a few of n's according to this procedure. Implement Jacobson's algorithm to choose the best one.

After many off-line implementations on the statistical data collected over IP, the results are positive and match Jacobson's recommended parameters.

The purpose of taking r in the form

$$r = \frac{2^n - 1}{2^n} \tag{24}$$

is again for the use of the integer shift operation.

Our method to choose the algorithm parameters depends only on the network characteristic, so it could be applied to other reliable transfer control protocol parameters.

5.4. Examples:

We would like to use an example to show the general principle of finding the parameters g and h. Figure 5.3 illustrates the comparison results of implementing Jacobson's algorithm using his recommended values, the Hurst parameter H, and our recommended values for the statistical round trip time data between ic13 in the Department of Electrical and Computer Engineering at the University of Manitoba and Boston.mci.net. 1000 packets were transmitted, each containing data of 100 bytes, 16 were lost, and $H = 0.788315$. The top three curves are timeout bounds using Jacobson's values, H, and our recommended values, respectively. The bottom curve is the sequence of the round-trip times of the transmitted packets. The average round trip time of the sample data is 121.929 (not including loss over IP). For Jacobson's values, the retransmission rate is 0.0356415 (not including loss over IP). The average timeout is 372.077 ms. For our recommendation, choosing N = 3, then $n = 3$, $r = 7/8$, $g = 1/8$, $h = g/2 = 1/16$ and the retransmission rate is 0.0305499 (not including loss over IP) and the average timeout 369.973 ms. If we take $r = H$, $g = 1 - H$, and $h = g/2$, then the retransmission rate is 0.0386965 (not including loss over IP) and the average timeout is 353.179 ms. These results are summarized in Table 5.1. Comparing the resulting values, our recommendation is slightly better (i.e., our recommendation reduces unnecessary retransmission to a small degree). The off-line implementations on a variety of different statistical data showed similar results.

Table 5.1 Summary

	Traditional RTO	H directly RTO	Recommended RTO
Retransmission rate	0.0356415	0.0386965	0.0305499
Average timeout	372.077 ms	353.179 ms	369.973 ms

For this example, taking $h = g/2$ is relatively optimized. Let us consider the relation $h = g \times a$ because in our recommendation $h = g/2$ and in Jacobson' recommendation $h = 2 \times g$. We want to know how the parameter a affects the retransmission rate and the average timeout.

Table 5.2 lists the retransmission rates (not including loss over IP) and the average timeouts for the above statistical data when $h = g \times a$ and a takes values ranging in the following set

$$\{\frac{1}{8}, \frac{1}{4}, \frac{3}{8}, \frac{1}{2}, \frac{5}{8}, \frac{3}{4}, \frac{7}{8}, 1, \frac{5}{4}, \frac{3}{2}, 1+\frac{5}{8}, 1+\frac{3}{4}, 1+\frac{7}{8}, 2\} \tag{25}$$

The corresponding plots are shown as (a) and (b), respectively, in Figure 5.4. The Jacobson's recommended value is the value as $a = 2$. From Figure 5.4, we can see that $a = 1/2$ is better.

Table 1:

a	1/8	1/4	3/8	1/2	5/8	3/4	7/8	1	5/4	3/2	13/8	7/4	15/8	2
retransmission rate	0.0336049	0.0346232	0.0325866	0.0305499	0.0336049	0.0325866	0.0336049	0.0325866	0.0336049	0.0325866	0.0336049	0.0346232	0.0346232	0.0356415
average timeout	361.2	367.2	369.1	370	370.5	370.9	371.2	371.4	371.7	371.8	371.9	372	372	372.1

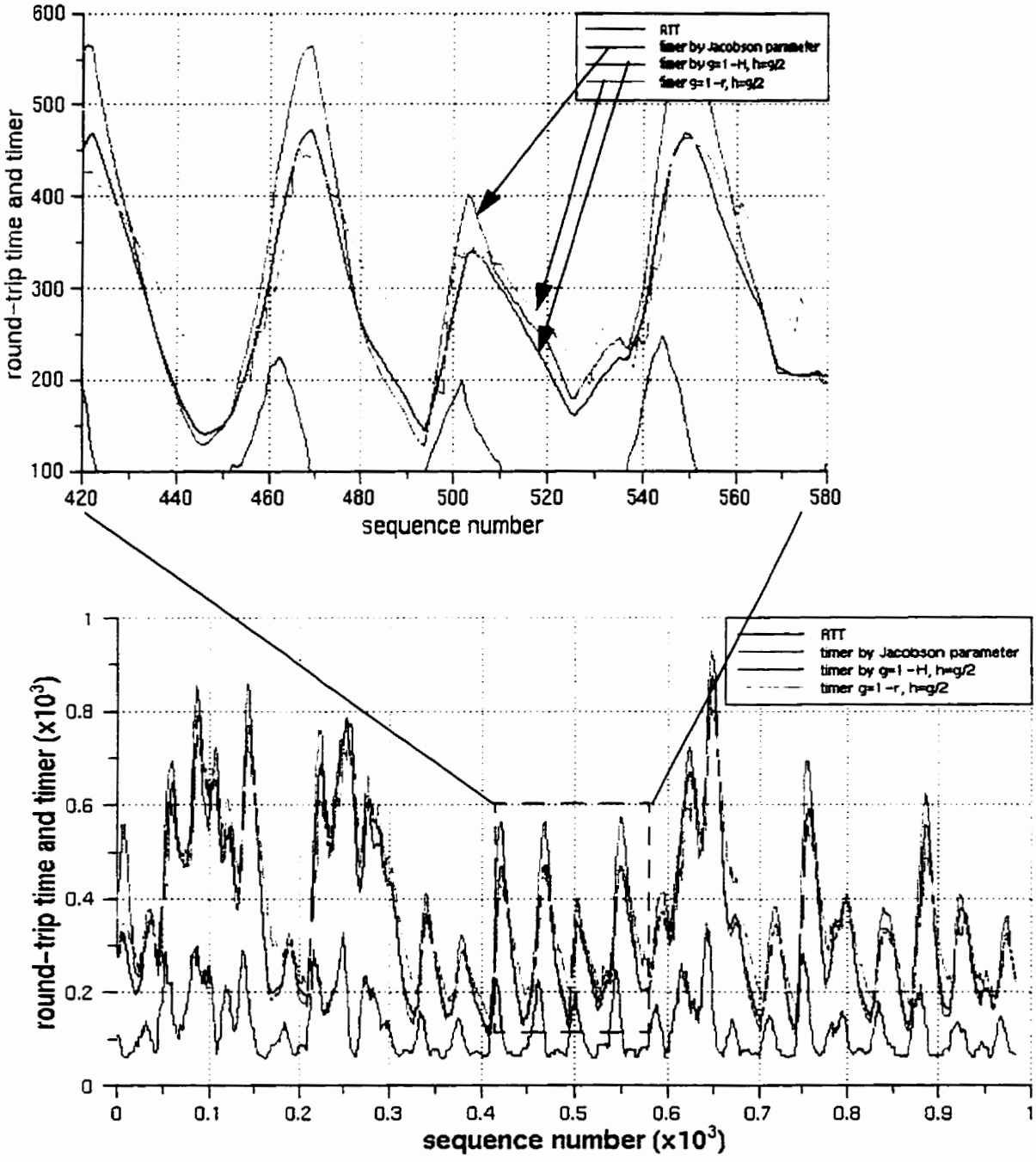
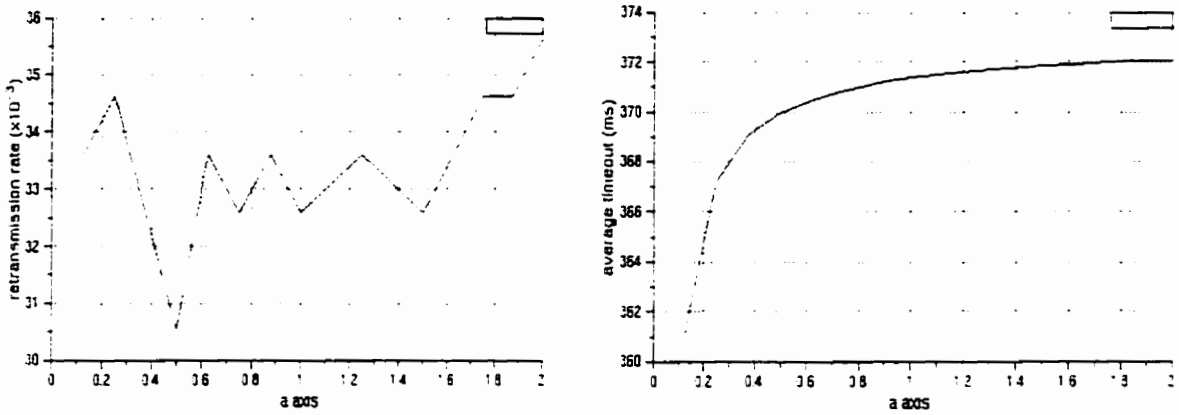


Figure 5.3 Comparison of three timers for a sequence of 1000 transmitted packets (fit in 10 points)



(a) (b)

Figure 5.4 Retransmission rate and average timeout with change of a

5.5. Summary

The distribution of the round-trip time over IP in the Internet has been illustrated to behave in a self-similar manner and hence can be characterized by the Hurst parameter. The Hurst parameter H was demonstrated to be useful in obtaining an estimate of the parameters in Jacobson's retransmission algorithm. Comparison of the implementations shows that the empirical estimate is positive. Since the Hurst parameter is a characteristic network property, the method is potentially applicable to other reliable transport algorithm designs which are related to the self-similarity. In fact, this is a preliminary result demonstrated through its application in RTO estimation. However, these results need further theoretical investigation to find the full usefulness of traffic descriptors such as the H parameter.

Chapter 6

Other Observations and Discussions

During measurements of the round trip time of the transmitted packet over IP in the Internet, we observed a lot of other characteristics worthy of consideration. These include a relationship between the loss rate and the Hurst parameter, the estimate of the number of nodes in the one way transmission route from the source to the sink, and the change of the loss rate over 24 hours.

6.1 Loss Rate and Hurst Parameter

We transmitted 1000 packets via pinging the routers atm1.manitoba.mbnet.mb.ca and borderx2-hssi2-0.Boston.mci.net, respectively, several times in different time intervals. The computing and measuring results are listed in the Table 6.1 and Table 6.2, respectively. The values in the last column indicate a possible relation between loss rate and the Hurst parameter H.

Table 2:

Test number	Loss rate	Hurst	(1-loss rate)+H
1	0.05	0.609649	1.605
2	0.06	0.616311	1.610
3	0.306	0.925364	1.616
4	0.23	0.645612	1.624

In this set of observation, the maximum loss rate is about 30%.

Table 3:

Test number	Loss rate	Hurst	(1-loss rate)+H
1	0.02	0.768512	1.767
2	0.03	0.732164	1.729
3	0.16	0.787424	1.771

These results lead to the following conjecture.

Conjecture. Given two hosts, there exists a constant number A ($1 < A < 2$) such that

$$(1 - \text{loss rate mean}) + \text{Hurst mean} = A. \tag{26}$$

6.2 On Estimation of the Number of Nodes

One may ask if the number of nodes between two hosts can be estimated. For our observations, it can be in some situations.

6.2.1 Theory

From probability theory, if T_n, $n = 1, 2,...$, are independent identically distributed exponential random variables having mean $1/\lambda$, $S_n = T_1+T_2+...+T_n$ has a gamma distribution with parameters n and λ, which in turn has the mean n/λ (ref. Section 3.3 in [22]):

$$\left((\lambda t)^{n-1} \cdot \lambda e^{-\lambda t} \right) / (n-1)! \tag{27}$$

where n = 1, 2, 3,.... Another quantity of interest is the arrival time of the *n*th event, called the *waiting time* until the *n*th event. Thus, if we assume that every router processes data with the same exponential distribution (it is an event), then the elapsed time of a packet from source to node (n+1) has the gamma distribution with the parameters n and λ (the packet is across n nodes). Let S_m be the propagation delay from the source to the (m+1)th node. Then we have the ratio of the mean times with respect to the number of nodes:

$$S_n / S_m = \frac{n}{m} \tag{28}$$

However, for the round trip time, it is difficult to find a similar relationship because we usually do not know the travel path of the packet (or response) for transmission back to the source. So the round trip path from the source to an intermediate node may not be a sub-path of the round trip path from the source to the destination. But we can use the above formula to estimate the number of nodes traversed (see the next section).

6.2.2 Testing in the Internet

For the topology of Figure 6.1, we have colleted seven sets of data for each node, each set data consisting of 1000 packets. For convenience, we use the following short-hand notations.

NOTATION:

- mbn: atm1.manitoba.mbnet.mb.ca;
- 238: 205.207.238.45;

- bboston: borderx2-hssi2-0.Boston.mci.net;
- seattle: bordercore1.Seattle.mci.net;
- seab: seabr2-gw.nwnet.net;
- wes: wes-core1.nwnet.net;
- nba: www.nba.com.

The mean time of round trip from the computer ic17 to the destination is shown in row two of the Table 6.3. The third row shows the value derived from the second row by formula: mean time/mbn, where mbn =10.526.

Table 4:

	mbn	238	bboston	seattle	seab	wes	nba
mean	10.5265	45.412	120.437	173.775	150.986	168.385	143.884
mean/mbn	1	4.3141	11.4413	16.5108	14.3434	15.996	13.6687

Referencing the one way path in Figure 6.1, let us consider the number m of nodes in the route from ic17.ee to node 238. By tracing the route, we know that the number n of nodes in the path from ic17.ee to node mbn is $n = 2$. From the last column in Table 6.3, the round trip time of the former is as about four times that of the latter. On average, the one way time of the former is two times that of the latter. According to the previous formula, m = 4, as expected.

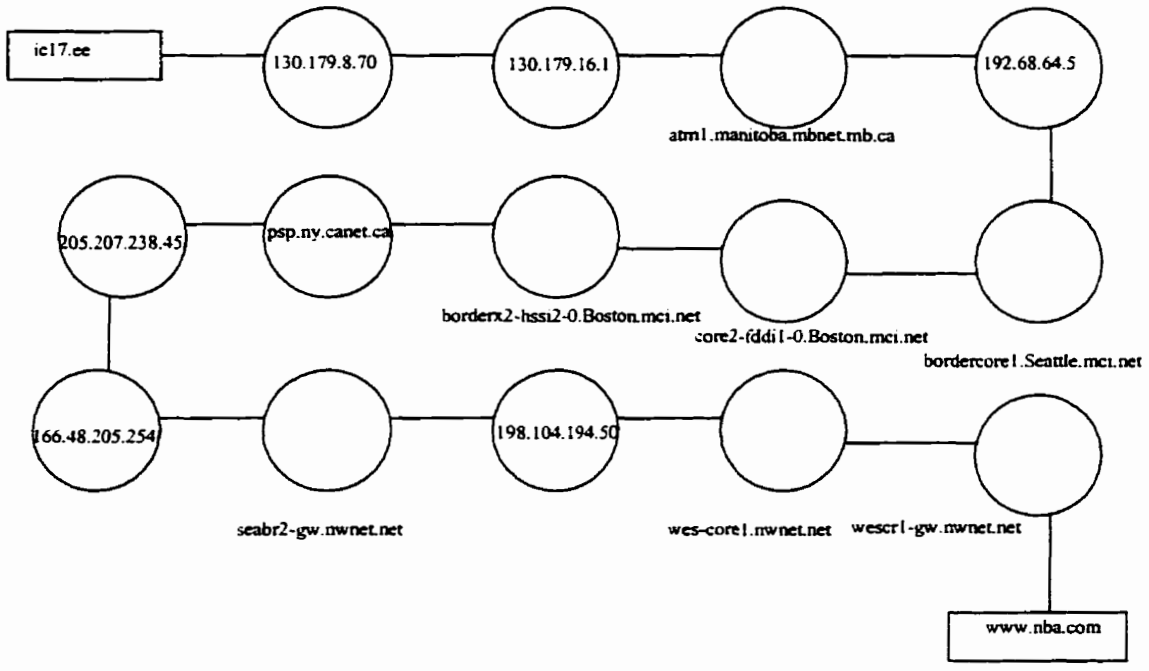


Figure 6.1 The test-bed of the one way route from ic17.ee.umanitoba.ca

This kind of estimation is also true for node nba and node mbn, but no others in Figure 6.1, since we can not simply divide the round trip time by two to get the one way time. Also note that some round trip delay to the intermediate node is longer than that to the destination. The main reason is that the number of nodes in the return (or response) path is greater and this path is different from the sending path and the assumption regarding the exponential service time at a router is likely not valid. For example, the number of nodes between node Seattle and ic17.ee is four times that between node mbn and ic17.ee (one way), but Seattle/mbn=16.5208. Therefore the formula $S_n/S_m = \frac{n}{m}$ does not hold in this case because of the above reason. For the same reason, the round trip delay to node Seattle is longer than that to node nba. The list of traceroute to each node is included in Appendix (1), and corresponds to Figure 6.1. Also we traced the routes between

ee.umanitoba.ca and cs1.gw.nts.uci.edu. The two direction routes are different (see Appendix (2)).

The theoretical and testing results for www.nba.com are consistent with respect to the number of routers as shown in Figure 6.2 (note that the statistical data has a heavy tail and phenomenon of aggregation), where n = 30 and $\lambda = 0.6756$ for the theoretical curve. The theoretical mean = standard mean + shift = $30/\lambda + 100 = 144$, which is almost the same as the statistical mean, 144.008. The Internet is complicated because there are a lot of unknowns such as different routing techniques, topology, capacity and so on.

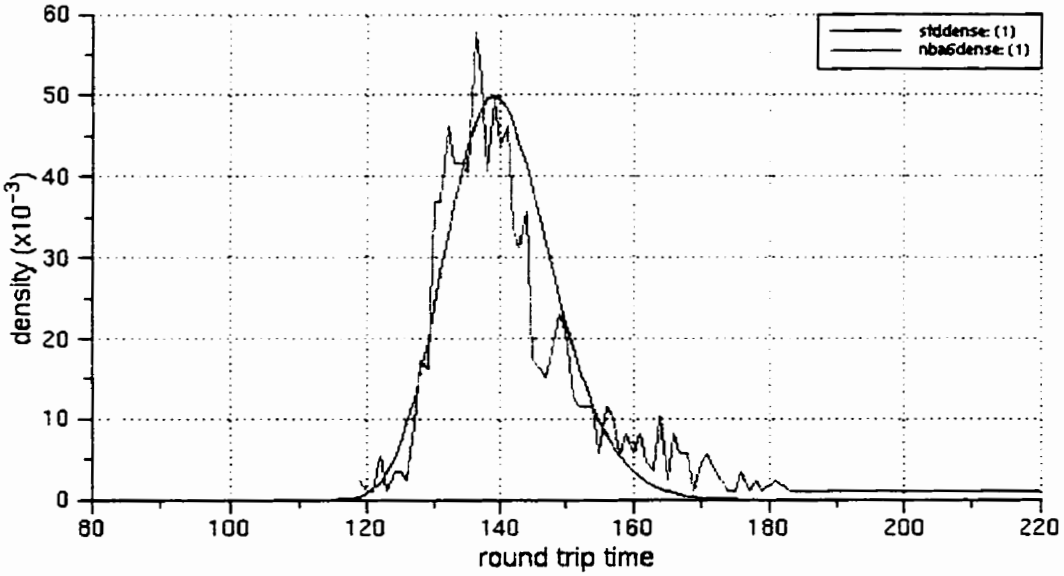


Figure 6.2 Comparison between theory and test

6.3 On 24 Hour Statistical Data of Loss Rate

Several 24-hour loss rate data over IP were collected. Figure 6.3 represents one of them for which we sent 1000 packets per hour to the University of California at Irvine (each packet contains 100 bytes of data). In Figure 6.3, we see that the loss rate is high between 12:00 noon and 18:00 pm, and low between 2:00am to 9:00am. You can see that at times the loss rate in the Internet is very high, for example, it can reach 25% as measured here.

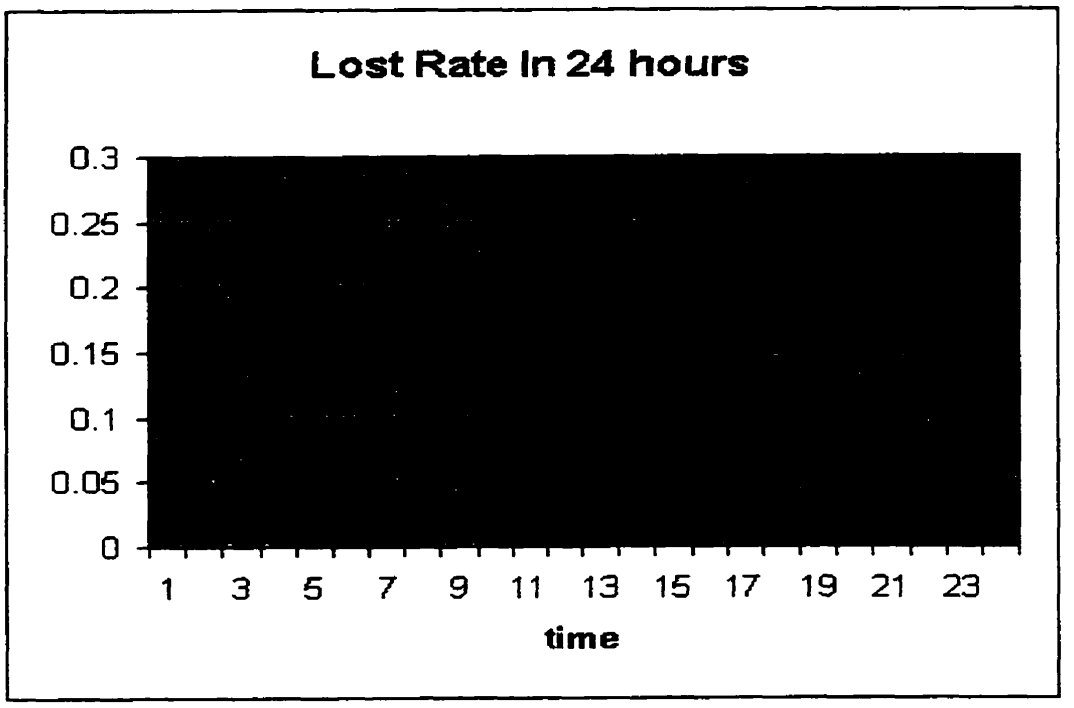


Figure 6.3 24-hour loss rate

Chapter 7

Conclusion

This thesis presented the application and development of place stochastic Petri nets to the modeling of communication networks. The modeling and analysis methods were shown to be applicable to the design of a deterministic reliable transmission protocol and the design of delay buffer for a real-time network. We believe that following the line described in this paper, fruitful results related to network design with timing constraints can be obtained.

In Chapter 5, we concentrated on the investigation about TCP/IP. We found that the distribution of round trip time of packet over IP in the Internet has the self-similarity property. The Hurst parameter H was then proposed to estimate the parameters in the Jacobson's retransmission algorithm. A comparison of the implementations shows that the estimate is positive. It is worthy to note that such a characteristics is a network property. The method we used could be potentially applicable to determine the parameters in other reliable transport protocol designs which are related to self-similarity. This demonstration was our primary purpose. However, these results need further theoretical investigation.

During our studies on the round trip time of the transmitted packet over IP in the Internet, we observed many interesting facts. Those were discussed in Chapter 6. We displayed a relationship between the loss rate and the Hurst parameter, and proposed a conjecture. The estimate of the number of nodes in the one way trip path was discussed. Those observations give us many challenging problems to consider and would be useful in addressing the timeout design presented in

Chapter 4 through chapter 6.

Appendix

(1) Trace routes from the source to every other nodes in the testbed

traceroute atm1.manitoba.mbnet.mb.ca
traceroute: Warning: ckecksums disabled
traceroute: Warning: Multiple interfaces found; using 130.179.8.102 @ le0
traceroute to atm1.manitoba.mbnet.mb.ca (204.112.54.161), 30 hops max, 40 byte packets
 1 130.179.8.70 (130.179.8.70) 2.829 ms 1.596 ms 1.511 ms
 2 atrouter.cc.umanitoba.ca (130.179.16.1) 2.256 ms 1.360 ms 1.375 ms
 3 atm1.manitoba.mbnet.mb.ca (204.112.54.161) 8.153 ms * 3.374 ms
traceroute 205.207.238.45
traceroute: Warning: ckecksums disabled
traceroute: Warning: Multiple interfaces found; using 130.179.8.102 @ le0
traceroute to 205.207.238.45 (205.207.238.45), 30 hops max, 40 byte packets
 1 130.179.8.70 (130.179.8.70) 2.650 ms 3.519 ms 1.602 ms
 2 atrouter.cc.umanitoba.ca (130.179.16.1) 2.152 ms 1.368 ms 1.477 ms
 3 atm1.manitoba.mbnet.mb.ca (204.112.54.161) 2.591 ms 2.216 ms 4.940 ms
 4 psp.mb.canet.ca (192.68.64.5) 5.104 ms 4.721 ms 9.356 ms
 5 205.207.238.45 (205.207.238.45) 41.766 ms * 46.409 ms
traceroute borderx2-hssi2-0.Boston.mci.net
traceroute: Warning: ckecksums disabled
traceroute to borderx2-hssi2-0.Boston.mci.net (204.70.179.117), 30 hops max, 40 byte packets
 1 130.179.8.70 (130.179.8.70) 2.678 ms 1.435 ms 1.661 ms
 2 atrouter.cc.umanitoba.ca (130.179.16.1) 2.603 ms 1.784 ms 26.767 ms
 3 atm1.manitoba.mbnet.mb.ca (204.112.54.161) 3.145 ms 2.116 ms 1.626 ms
 4 psp.mb.canet.ca (192.68.64.5) 13.938 ms 28.951 ms 24.799 ms
 5 205.207.238.45 (205.207.238.45) 32.119 ms 40.766 ms 53.956 ms
 6 psp.ny.canet.ca (205.207.238.154) 44.761 ms 251.832 ms 115.158 ms
 7 borderx2-hssi2-0.Boston.mci.net (204.70.179.117) 67.039 ms * 46.777 ms

traceroute bordercore1.Seattle.mci.net
traceroute: Warning: ckecksums disabled
traceroute: Warning: Multiple interfaces found; using 130.179.8.102 @ le0
traceroute to bordercore1.Seattle.mci.net (166.48.204.1), 30 hops max, 40 byte packets
 1 130.179.8.70 (130.179.8.70) 2.716 ms 1.605 ms 1.474 ms
 2 atrouter.cc.umanitoba.ca (130.179.16.1) 2.367 ms 1.471 ms 1.479 ms
 3 atm1.manitoba.mbnet.mb.ca (204.112.54.161) 3.140 ms 2.201 ms 2.028 ms
 4 psp.mb.canet.ca (192.68.64.5) 7.789 ms 5.493 ms 4.533 ms
 5 205.207.238.45 (205.207.238.45) 45.943 ms 33.942 ms 35.071 ms
 6 psp.ny.canet.ca (205.207.238.154) 40.208 ms 48.968 ms 49.686 ms
 7 borderx2-hssi2-0.Boston.mci.net (204.70.179.117) 82.133 ms 275.873 ms 158.607 ms
 8 core2-fddi1-0.Boston.mci.net (204.70.179.65) 273.687 ms 131.934 ms 80.296 ms
 9 bordercore1.Seattle.mci.net (166.48.204.1) 205.456 ms * 150.390 ms
traceroute seabr2-gw.nwnet.net

traceroute: Warning: ckecksums disabled
traceroute: Warning: seabr2-gw.nwnet.net has multiple addresses; using 204.200.240.65
traceroute: Warning: Multiple interfaces found; using 130.179.8.102 @ le0
traceroute to seabr2-gw.nwnet.net (204.200.240.65), 30 hops max, 40 byte packets
1 130.179.8.70 (130.179.8.70) 2.871 ms 1.601 ms 1.406 ms
2 atrouter.cc.umanitoba.ca (130.179.16.1) 2.014 ms 1.706 ms 1.467 ms
3 atm1.manitoba.mbnet.mb.ca (204.112.54.161) 3.981 ms 2.041 ms 2.009 ms
4 psp.mb.canet.ca (192.68.64.5) 14.160 ms 22.344 ms 8.248 ms
5 205.207.238.45 (205.207.238.45) 41.283 ms 40.403 ms 35.376 ms
6 psp.ny.canet.ca (205.207.238.154) 63.233 ms 40.875 ms 45.184 ms
7 borderx2-hssi2-0.Boston.mci.net (204.70.179.117) 65.865 ms 94.265 ms 358.782 ms
8 * core2-fddi1-0.Boston.mci.net (204.70.179.65) 76.988 ms 62.352 ms
9 * bordercore1.Seattle.mci.net (166.48.204.1) 130.877 ms 147.016 ms
10 166.48.205.254 (166.48.205.254) 394.756 ms 137.536 ms *
11 seabr2-gw.nwnet.net (204.200.9.6) 163.891 ms * 136.114 ms
traceroute wes-core1.nwnet.net
traceroute: Warning: ckecksums disabled
traceroute: Warning: wes-core1.nwnet.net has multiple addresses; using 204.202.45.209
traceroute: Warning: Multiple interfaces found; using 130.179.8.102 @ le0
traceroute to wes-core1.nwnet.net (204.202.45.209), 30 hops max, 40 byte packets
1 130.179.8.70 (130.179.8.70) 2.812 ms 9.245 ms 1.485 ms
2 atrouter.cc.umanitoba.ca (130.179.16.1) 2.813 ms 1.475 ms 1.332 ms
3 atm1.manitoba.mbnet.mb.ca (204.112.54.161) 4.145 ms 5.541 ms 31.043 ms
4 psp.mb.canet.ca (192.68.64.5) 31.552 ms 17.823 ms 17.851 ms
5 205.207.238.45 (205.207.238.45) 93.678 ms 61.557 ms 47.416 ms
6 psp.ny.canet.ca (205.207.238.154) 77.897 ms 278.503 ms 265.137 ms
7 borderx2-hssi2-0.Boston.mci.net (204.70.179.117) 324.950 ms 227.441 ms 247.910 ms
8 core2-fddi1-0.Boston.mci.net (204.70.179.65) 225.090 ms 82.778 ms 70.885 ms
9 bordercore1.Seattle.mci.net (166.48.204.1) 141.745 ms 142.829 ms 140.652 ms
10 166.48.205.254 (166.48.205.254) 142.684 ms 144.169 ms 137.565 ms
11 seabr2-gw.nwnet.net (204.200.9.6) 136.748 ms 138.058 ms 138.895 ms
12 * 198.104.194.50 (198.104.194.50) 135.051 ms 135.268 ms
13 * * wes-core1.nwnet.net (198.104.194.46) 347.384 ms

(2) Trace two direction routes

traceroute www.ee.umanitoba.ca from cs1.gw.nts.uci.edu

1 cs1.gw.nts.uci.edu (128.200.38.1) 2 ms 1 ms 1 ms 2
dimrill.gw.nts.uci.edu (128.200.245.20) 1 ms 1 ms 1 ms
3 128.200.202.2 (128.200.202.2) 2 ms 2 ms 2 ms
4 sl-gw8-ana-10-0-T3.sprintlink.net (144.228.170.5) 3 ms 4 ms 3 ms
5 sl-bb2-ana-6-0-0.sprintlink.net (144.228.70.12) 4 ms 5 ms 5 ms
6 core3-hssi3-0.Bloomington.mci.net (206.157.77.41) 276 ms 21 ms 24
ms

7 core2.Boston.mci.net (204.70.4.237) 77 ms 75 ms 88 ms
8 borderx2-fddi-1.Boston.mci.net (204.70.179.68) 213 ms 239 ms 230
ms
9 canet.Boston.mci.net (204.70.179.118) 101 ms * *
10 border1-hssi6-0.quebec.canet.ca (205.207.238.153) 112 ms * 109 ms
11 psp2.mb.canet.ca (205.207.238.46) 137 ms 133 ms 155 ms
12 regional1.mb.canet.ca (192.68.64.101) 146 ms 139 ms 150 ms
13 atrouter.cc.umanitoba.ca (204.112.54.162) 148 ms 148 ms 141 ms
14 bbrouter.cc.umanitoba.ca (130.179.16.210) 152 ms 153 ms 144 ms
15 ic12.ee.umanitoba.ca (130.179.8.48) 145 ms * 149 ms

traceroute cs1.gw.nts.uci.edu from www.ee.umanitoba.ca
traceroute: Warning: ckecksums disabled
traceroute: Warning: cs1.gw.nts.uci.edu has multiple addresses; using 128.195.1.61
traceroute to cs1.gw.nts.uci.edu (128.195.1.61), 30 hops max, 40 byte packets
1 130.179.8.70 (130.179.8.70) 3.082 ms 1.892 ms 1.886 ms
2 atrouter.cc.umanitoba.ca (130.179.16.1) 2.796 ms 5.164 ms 1.974 ms
3 atm1.manitoba.mbnet.mb.ca (204.112.54.161) 4.198 ms 14.922 ms 17.470 ms
4 psp.mb.canet.ca (192.68.64.5) 45.750 ms 39.753 ms 38.512 ms
5 205.207.238.45 (205.207.238.45) 63.334 ms 57.485 ms 63.754 ms
6 psp.ny.canet.ca (205.207.238.154) 75.805 ms 81.822 ms 72.705 ms
7 borderx2-hssi2-0.Boston.mci.net (204.70.179.117) 247.673 ms 63.110 ms 56.842 ms
8 * core2-fddi1-0.Boston.mci.net (204.70.179.65) 62.384 ms 61.764 ms
9 core4.WestOrange.mci.net (204.70.4.77) 85.739 ms 76.613 ms 68.334 ms
10 somerouter.sprintlink.net (206.157.77.106) 87.265 ms 101.590 ms 90.249 ms
11 sl-bb10-pen-1-2.sprintlink.net (144.232.5.45) 85.890 ms 94.985 ms 78.501 ms
12 sl-bb2-fw-5-0-0-155M.sprintlink.net (144.232.8.157) 148.050 ms 158.963 ms 149.952 ms
13 sl-bb2-fw-0-0-0-155M.sprintlink.net (144.232.1.138) 151.797 ms * 145.784 ms
14 sl-bb11-ana-1-0.sprintlink.net (144.232.8.50) 144.581 ms 146.037 ms 137.398 ms
15 sl-bb1-ana-4-0-0-155M.sprintlink.net (144.232.1.78) 184.040 ms * 150.525 ms
16 sl-gw8-ana-0-0.sprintlink.net (144.228.70.10) 153.006 ms * 141.062 ms
17 * sl-ucirvine-1-0-T3.sprintlink.net (144.228.170.6) 136.933 ms 142.062 ms
18 dimrill.gw.nts.uci.edu (128.200.202.1) 153.447 ms 144.383 ms 166.487 ms
19 * * *
20 cs1.gw.nts.uci.edu (128.200.245.15) 146.216 ms * 142.554 ms

References

[1] R. E. Barlow and A. S. Wu, *Coherent Systems with Multi-State Components, Mathematics of Operations research*, Vol. 3, No. 4, November 1978.

[2] L. A. Baxter, *Continuum Structure I*, J. Appl. prob. 21, pp. 802--815, 1984.

[3] H. W. Block and T. H. Savits, *A decomposition for Multistate Monotone Systems*, J. Appl. Prob. 19, pp. 391--402, 1982.

[4] M. E. Crovella and A. Bestavros, *Self-similarity in World Wide Web Traffic: evidence and possible causes*, in Proc. 1996 ACM SIGMETRICS. Int. Conf. Meas. Model. Comput. Syst., May 1996.

[5] A. Erramilli, O. Narayan, and W. Willinger, *Experimental Queueing Analysis with Long-range Dependent Packet Traffic*, pp. 209--223, IEEE/ACM transactions on Networking, April 1996.

[6] M. A. Escalante and N. J. Dimopoulos, *A Probabilistic Timing Analysis for Synthesis in Microprocessor Interface Design*, Manuscript, 1997, Department of Electrical and Computer Engineering, University of Victoria.

[7] M. A. Escalante and N. J. Dimopoulos, *Assessing the Feasibility of Interface Designs before their Implementation*, Manuscript, 1997, Department of Electrical and Computer Engineering, University of Victoria.

[8] M. A. Escalante and N. J. Dimopoulos, *Timing analysis for synthesis in microprocessor interface design*, Proc. Seventh High-level Synthesis Symposium, pp. 23-28, May 1994.

[9] G. Gratzer, *Boolean Functions on Distributive Lattices*, Acta Math. Acad. Sci. Hungar. 15, pp. 195--201, 1964.

[10] G. Gratzer, *General Lattice Theory*, Pure and Applied Mathematics, Vol. 75, Academic Press Inc., 1978.

[11] W. S. Griffith, *Multistate Reliability Models*, J. Appl. Prob. 17, 735--744, 1980.

[12] V. Jacobson, *Congestion Avoidance and Control*, Proc. SIGCOMM '88 conf., ACM, pp. 314-329, 1988.

[13] S. Jamin, P. B. Danzig, S. Shenker, and L. Zhang, *A measurement-based admission control algorithm for integrated services packet networks*, ACM/SIGCOMM Comput. Commun. Rev., Vol25, pp. 2--13, 1995.

[14] L. M. Leemis, *Reliability: probability models and statistical methods*, Printice-Hall, 1995.

[15] S. Lei, D. Wang, D. C. Blight, and R. D. McLeod, *Reliable Network Design: Characterization of Structure Function, Manuscript*, 1998.

[16] W. Leland, M. Taqqu, W. Willinger, and D. Wilson, *On the Self-similar Nature of Ethernet Traffic (extended Version)*, IEEE/ACM Transactions on Networking, February 1994.

[17] M. K. Molloy, *Performance analysis using stochastic Petri nets*, IEEE Trans. Computers, Vol. C-31, No. 9, pp. 913--917, September, 1982.

[18] M. K. Molloy, *Discrete time stochastic Petri nets, IEEE Trans. Software Eng.*, Vol. SE-11, No. 4, pp. 417--423, April 1985.

[19] T. Murata, *Petri Nets: Properties, Analysis and Applications*, Proceedings of IEEE, Vol. 77, No. 4, April 1989.

[20] S. Petri, Kommunikation mit Automaten, Ph. D. thesis, Institut fur Instrumentelle Mathematik, Schriffen des IIM, Germany, 1962.

[21] V. Paxson and S. Floyd, *Wide Area Traffic: The failure of Poisson Modeling*, IEEE/ACM Transactions on Networking, June 1995.

[22] S. M. Ross, *Introduction to Probability Models*, 4th edition, Academic Press, Inc., 1989.

[23] H. Schulzrinne, S. Casner, R. Frederick & V. Jacobson, *RTP: A Transport Protocol for Real-Time Applications*, RFC 1889, January 1996.

[24] M. Schwartz, *Telecommunication Networks: Protocol, Modeling, and Analysis*, Addison-Wesley Publishing Company, 1987.

[25] D. R. Shier, *Network Reliability and Algebraic Structures*, Clarendon Press, 1991.

[26] J. D. Spragins, J. C. Sinclair, Y. J. Kang, and H. Jafari, *Current Telecommunication Network Reliability Models: A Critical Assessment*, IEEE Journal on Selected Areas in Communications, Vol. SAC-4, No. 7, October, 1986.

[27] W. Stallings, *High-speed networks: TCP/IP and ATM design principles*, Prentice-Hall, 1998.

[28] W. R. Stevens, *TCP/IP Illustrated Volume 1: The protocols*, Addison Wesley Longman Inc., 1994.

[29] A. S. Tanenbaum, *Computer Networks*, 3rd Edition, Printice Hall PTR, 1996.

[30] D. Wang, D. C. Blight, and R. D. McLeod, *Self-Similarity of the Round trip Time and the Jacobson's Retransmission Control Algorithm*, Manuscript, 1998.

[31] D. Wang, D. C. Blight, and R. D. McLeod, *A Probabilistic Timing Analysis for Synthesis in*

Reliable Transmission Design, Manuscript, 1998.

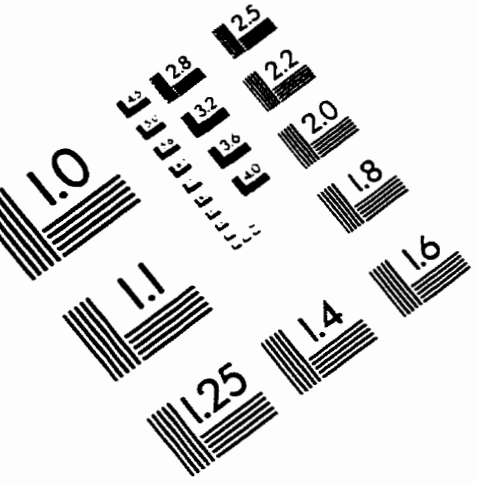
[32] W. Willinger, M. S. Taqqu, R. Sherman, and D. V. Wilson, *Self-Similarity Through High-Variability: Statistical Analysis of Ethernet LAN Traffic at the Source Level*, IEEE/ACM Transactions on Networking, Vol 5, No. 1, February 1997.

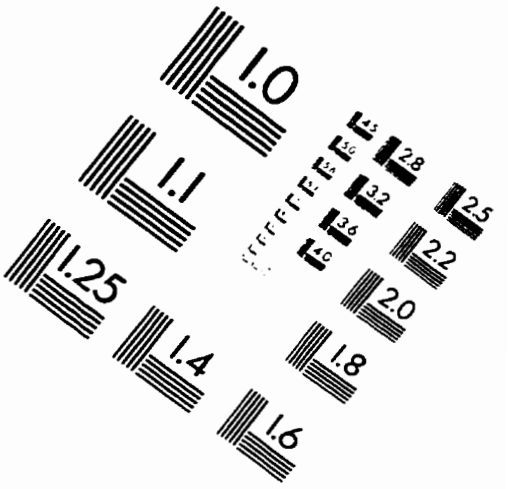
[33] W. Willinger, D. Wilson, and M. Taqqu, *Self-similar Traffic Modeling for High-speed Networks*, ConneXions, November 1994.

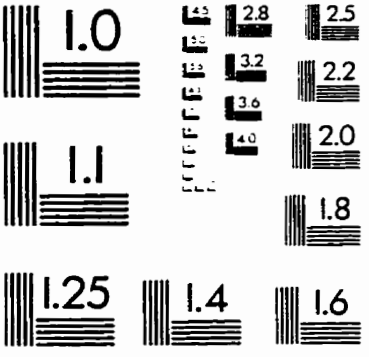
[34] S. Zacks, *Introduction to Reliability Analysis: probability models and statistical methods*, Springer-Verlag, 1992.

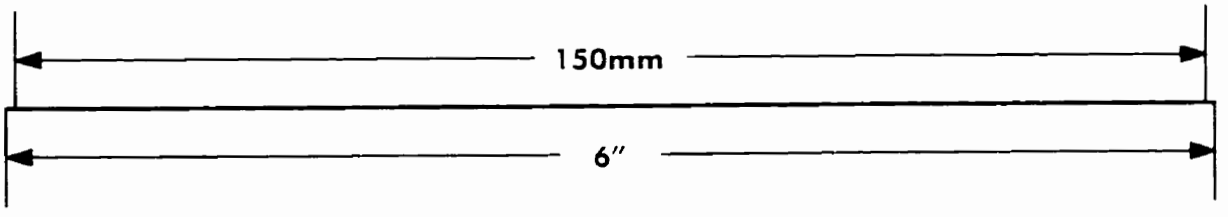
[35] I. Norros, *On the Use of Fractional Brownian Motion in the Theory of Connectionless networks*, IEEE Journal on Selected Areas in Communications, August 1995.

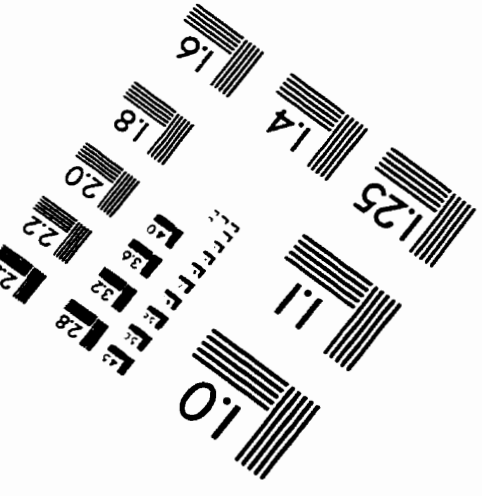
IMAGE EVALUATION
TEST TARGET (QA-3)

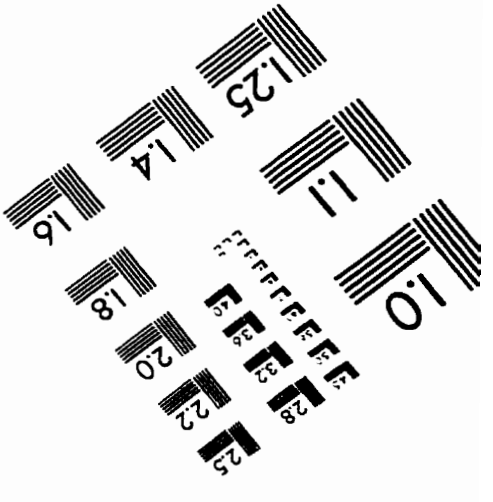












APPLIED IMAGE, Inc
1653 East Main Street
Rochester, NY 14609 USA
Phone: 716/482-0300
Fax: 716/288-5989