Modeling TCP/IP Software Implementation Performance And Its Application For Software Routers

By Oscar Iván Lepe Aldama

Dr.	
President	

Dr.	
Secretari	

Dr.	
Vocal	

Dr.	
Vocal	

Dr.	
Vocal	

Data de la defensa pública	
Qualificació	

Contents

List of figures

List of tables

Preface

Dedico esta obra a mi esposa, Tania Patricia, y a nuestros hijos, Pedro Darío y Sebastián.

Al mismo tiempo, quiero agradecer a Tania y a Pedro su apoyo y paciencia durante el tiempo que estuvimos fuera de casa, viviendo no en las mejores condiciones. Sólo espero que esta precariedad haya sido mitigada por lo enriquecedor de las experiencias que hemos vivido, que siento nos harán mejores personas al habernos recalcado la importancia de la unidad familiar, y el valor de nuestras costumbres, nuestra gente y nuestra tierra.

Agradezco a mi director de tesis, Jorge García-Vidal, por su invaluable tutela y dedicación. Difícilmente olvidaré las incontables y largas reuniones de discusión que le dieron vida a esta obra, ni las numerosas noches en vela que pasamos ultimando los detalles de los artículos que enviamos a revisión. Pero también difícilmente olvidaré las tardes de café o las tardes en el parque con nuestros hijos, donde hablábamos de todo menos del trabajo.

Agradezco al pueblo de México, que con su esfuerzo permite la existencia de programas de becas para que mexicanos hagan estudios de postgrado en el extranjero, como el programa gestionado por el Consejo Nacional de Ciencia y Tecnología, o el gestionado por el Centro de Investigación Científica y de Educación Superior de Ensenada, que me dieron cobijo.

Y en general agradezco a todas las personas que de forma directa o indirecta me ayudaron a la consecución de esta obra, por ejemplo a los profesores del DAC/UPC y a los revisores anónimos en los congresos. En particular quiero mencionar, en orden alfabético, a Alejandro Limón Padilla, David Hilario Covarrubias Rosales, Delia White, Eva Angelina Robles Sánchez, Francisco Javier Mendieta Jiménez, Jorge Enrique Preciado Velasco, José María Barceló Ordinas, Llorenç Cerdà Alabern, Olga María Casals Torres, Oscar Alberto Lepe Peralta, Victoriano Pagoaga. A todos ellos muchas gracias.

Chapter 1

Introduction

1.1 Motivation

Evidently, computer networks and the computer applications than run over them are fundamental to today's human society. Indeed, some kind of these telematic systems is fundamental for the proper operation of the New York Stock Exchange, for instance, but also some other kind is fundamental for providing telephony service to little villages in hard-to-reach places, like the numerous villages at the mountains of Chiapas, México. As it is well known, telematic technology's application for better human society is only limited by humans' imagination.

Today's human's necessities for information processing and transportation present complex problems. For solving these problems, scientists and engineers are required to produced ideas and products with an ever-increasing number of components and inter-components relationships. To cope with this complexity, developers invented the concept of layered architectures, which allow a structured "divide to conquer" approach for designing complex systems by providing a step-by-step enhancement of system services.

Unfortunately, layered structures can result in relatively low performance telematic systems—and often they are—if implemented carelessly [Clark 1982; Tennenhouse 1989; Abbott and Peterson 1993; Mogul and Ramakrishnan 1999]. To understand this, let us consider the following:

- Telematic systems are mostly implemented with software.
- Each software layer is designed as an independent entity that concurrently and asynchronously communicates with its neighbors through a message-passing interface. This allows for better interoperability, manageability and extensibility.
- In order to allow software layers to work concurrently and asynchronously, the host computer system has to provide a secure and reliable multiprogramming environment through an operating system.
- Ideally, the operating system should perform its role without consuming a significant share of processing resources. Unfortunately, as reported elsewhere [Druschel 1996], current operating systems are threatening to become bottlenecks when processing input/output data streams. Moreover, they are the source of statistical delays—incurred as each data unit is marshaled through the layered software—that hamper the correct deployment of important services.

Others have recognized this problem and have conducted studies analyzing some aspects of some operating systems' computer networks software, networking software for short. For us it is striking that although these studies are numerous, (a search through the ACM's Digital Library on the term "(c.2 and d.4 and c.5)<IN>ccs" returns 54 references, and a search through IEEE's Xplore on the term "'protocol processing'<IN>de" returns 81 references) only one of them pursued to build a general model of the networking software—see this chapter's section on "related work". Indeed, although different systems' networking software has more similarities than differences, as we will later discuss, most of these studies have only focused in identifying and solving particular problems of particular systems. When saying this we do not denied the importance of this work, however, we believe that modeling is an important part of doing research.

The Internet protocol suite (TCP/IP) [Stevens 1994] is, nowadays, the preferred technology for networking. Of all possible implementations, the one done at the University of California at Berkeley for the Berkeley Software Distribution operating system, or BSD, has been used as the starting point for most available systems. The BSD operating system [McKusick et al. 1996], which is a flavor of the UNIX system [Ritchie and Thompson 1978], was selected as the foundation for implementing the first TCP/IP suite back in the 80's.

1.2 Scope

Naturally, for modeling a system high degrees of observability and controllability are required. For us this means free access to both networking software's source code and host computer's technical specifications. (When we speak of free, we are referring to freedom, not price.) Today's personal computer (PC) technology provides this. Indeed, not only there is tons of freely available detailed technical documentation on Intel's IA32 PC hardware but also there are several PC operating systems with open source policy. Of those we decide to work with FreeBSD, a 4.4BSD-Lite derived operating system optimized to be run on Intel's IA32 PCs.

When searching for a networking application for which the appliance of a performance model could be of importance we found software routers. A software router can be defined as a general-purpose computer that executes a computer program capable of forwarding IP datagrams among network interface cards attached to its input/output bus (I/O bus). Evidently, software routers have performance limitations because they use a single central processing unit (CPU) and a single shared bus to process all packets. However due to the ease with which they can be programmed for supporting new functionality—securing communications, shaping traffic, supporting mobility, translating network addresses, supporting applications' proxies, and performing n-level routing—software routers are important at the edge of the Internet.

1.3 Dissertation's thesis

From all the above we came up with the following dissertation thesis:

Is it possible to build a queuing network model of the Internet protocols' BSD implementation that can be used for predicting with reasonable accuracy not only the mean values of the operational parameters studied but also their cumulative probability function? And, can this model be applied for studying the performance of PC-based software routers supporting communication quality assurance mechanisms, or Quality-of-Service (QoS) mechanisms?

1.4 Synopsis

Three are the main contributions of this work. In no particular order:

- A detailed performance study of the software implementation of the TCP/IP protocols suite, when executed as part of the kernel of a BSD operating system over generic PC hardware
- A validated queuing network model of the studied system, solved by computer simulation
- An I/O bus utilization guard mechanism for improving the performance of software routers supporting QoS mechanisms and built upon PC hardware and software

This document presents our experiences building a performance model of a PC-based software router. The resulting model is an open multiclass priority network of queues that we solved by simulation. While the model is not particularly novel from the system modeling point of view, in our opinion, it is an interesting result to show that such a model can estimate, with high accuracy, not just average performance-numbers but the complete probability distribution function of packet latency, allowing performance analysis at several levels of detail. The validity and accuracy of the multiclass model has been established by contrasting its packet latency predictions in both, time and probability spaces. Moreover, we introduced into the validation analysis the predictions of a router's single queue model. We did this for quantitatively assessing the advantages of the more complex multiclass model with respect to the simpler and widely used but not so accurate, as here shown, single queue model, under the considered scenario that the router's CPU is the system bottleneck and not the communications links. The single queue model was also solved by simulation.

Besides, this document addresses the problem of resource sharing in PC-based software routers supporting QoS mechanisms. Others have put forward solutions that are focused on suitably distributing the workload of the CPU—see this chapter's section on "related work". However, the increase in CPU speed in relation to that of the I/O bus—as here shown—means attention must be paid to the effect the limitations imposed by this bus on the system's overall performance. We propose a mechanism that jointly controls both I/O bus and CPU operation. This mechanism involves changes to the operating system kernel code and assumes the existence of certain network interface card's functions, although it does not require changes to the PC hardware. A performance study is shown that provides insight into the problem and helps to evaluate both the effectiveness of our approach, and several software router design trade-offs.

1.5 Outline

The rest of this chapter is devoted to discuss about related work. Chapter 2's objective is to understand the influence that operating system design and implementation technique have over the performance of the Internet protocol's BSD software implementation. Chapter 3 presents our experiences building, validating and conducting the parameterization of a performance model of a PC-based software router. Moreover, it presents some results from applying the model for capacity planning. Chapter 4 addresses the problem of resource sharing in PC-based software routers supporting communication quality assurance mechanisms. Furthermore, it presents our mechanism for jointly controlling the router's CPU and I/O bus, indispensable for a software router to support communication quality assurance mechanisms.

1.6 Related work

Cabrera et al. [1988] (in reality, they presented their study's results on July, 1985) is the earliest work we found across the publicly accessible literature on TCP/IP's implementation experimental evaluation. Their study was an ambitious one, which objective was to assess the impact that different processors, network hardware interfaces, and Ethernets have on the communication across machines, under various hosts' and communication medias' load conditions. Their measurements highlighted the ultimate bounds on communication performance perceived by application programs. Moreover, they presented a detailed timing analysis of the dynamic behavior of the networking software. They studied TCP/IP's implementation within 4.2BSD when ran by then state of the art minicomputers, attached to legacy Ethernets. Consequently, their study's results are no longer valid. Worst yet, they used the `gprof(1)` and `kgmon(8)` tools for profiling. These tools are only supported by software and, consequently, produce results that have limited accuracy when compare to results produce by performance monitoring hardware counters, as we do. However complete their study was, they did not pursue to build a system model, like we do.

Sanghi et al. [1990] is the earliest work we found across the publicly accessible literature on TCP/IP's implementation experimental evaluation that uses software profiling. Their study is very narrow when compare to ours in the sense that their study's objective was only to determine the suitability of roundtrip time estimators for TCP implementations.

Papadopoulos and Gurudatta [1993] presented results of a more general study on TCP/IP's implementation performance, obtained using software profiling. They studied TCP/IP's implementation within a 4.3BSD-derived operating system—SUN OS. Like our work, their study's objective was to characterize the performance of the networking software. Conversely, their study was not aimed to produce a system model.

Kay and Pasquale [1996] (in reality, they presented their study's results on September, 1994) conducted another TCP/IP's implementation performance study. Differently from previous work, their study was carried out at a different granularity level; that is, they went inside the networking functions and analyzed how these functions used the source code—touching data, protocol-specific processing, memory buffers manipulation, error checking, data structure manipulation, and operating system overhead. Moreover, their study's objective was to guide a search for bottlenecks in archiving high throughput and low latency. Once again, they did not pursue building a system model.

Ramamurthy [1988] modeled the UNIX system using a queuing network model. However, his model characterizes the system's multitasking properties and therefore cannot be applied to study the networking software, which is governed by the software interrupt mechanism. Moreover, Ramamurthy's model was only solved for predicting mean values, something that is not enough when conducting performance analyses of today's networking software. What is required is a model capable of producing the complete probability function of operational parameters so analysis at several levels of detail may be performed.

Björkman and Gunningberg [1998] modeled an Internet protocols' implementation using a queuing network model, however, their model characterizes a user-space implementation (base on a parallelized version of University of Arizona's x-kernel [Hutchinson and Peterson 1991]) and therefore disregards the software interrupt mechanism. Moreover, their model was aimed to predict only system throughput (measured in packets per second) when the protocols are hosted by shared-memory multiprocessor systems. Besides, their study was aimed only for high-performance distributed computing, where it is considered that connections are always open with a steady stream of messages—that is, no retransmissions or other unusual events occur. All this impedes the utilization of their model for our intended applications.

Packet service disciplines and their associated performance issues have been widely studied in the context of bandwidth scheduling in packet-switched networks [Zhang 1995]. Arguably, several such disciplines now exist that are both fair—in terms of assuring access to link bandwidth in the presence of competing packets flows—and easy to implement efficiently—with both hardware and software. Recently, an interest has arisen to map these packet service disciplines in the context of CPU scheduling for programmable and software routers. Qie et al. [2001], and Chen and Morris [2001], for a software router, and Pappu and Wolf [2001], for a programmable router, have put forward solutions that are focused on suitably distributing the workload of a router's processor. However, neither the formers nor the latters consider the problem of I/O bus bandwidth scheduling. And this problem is important in the context of software routers, as we demonstrate.

Chiueh and Pradhan [2000] recognized the suitability and inherent limitations of using PC technology for implementing software routers. In order to overcome the performance limitations of PCs' I/O buses, and to construct high-speed routers, they have proposed using clusters of PCs. Upon this architecture, several PCs having at most one network interface card each are tightly couple by means of a Myrinet system area network to conform a software router with as many subnetwork attachments as computing nodes. After solving all internodes communication, memory coherency and routing table distribution problems—arguably not an easy task—a "clustered router" may be able to overcome the limitations of current PCs' I/O buses and not only provide high performance—in term of packet per second—but also support QoS mechanisms. Our work, however, is aimed at supporting QoS mechanisms in clearly simpler and less expensive PC based software routers.

Scottis, Krunz and Liu [1999] recognized the performance limitations of the omnipresent Peripheral Component Interconnect (PCI) PC I/O bus to support QoS mechanisms, and proposed an enhancement. Conversely to our software enhancement, theirs introduced a new bus arbitration protocol that has to be implemented with hardware. Moreover, their bus arbitration protocol is aimed to improve bus support for periodic and predictable real-time streams only, and clearly this is not suitable for Internet routers.

There is general agreement in the PC industry that the demands of current user applications are quickly overwhelming the shared parallel architecture and limited bandwidth of the various types of PCI buses. (Erlanger, L. Editor "High-performance busses and interconnects," http://www.pcmag.com/article2/0,4149,293663,00.asp current as 8 July 2002.) With this increasing demand and its lack of QoS, PCI has been due for a replacement in different system and application scenarios for more than a few years now. 3GIO, InfiniBand, RapidIO and HyperTransport are new technologies designed to improve I/O and device-to-device performance in a variety of system and application categories, however, not all are direct replacements for PCI. In this sense, 3GIO and InfiniBand may be considered as direct candidates to succeed PCI. All these technologies have in common that they define packet-based, point-to-point serial connections with layered communications, which readily support QoS mechanisms. However, due to the large installation base of PCI equipment, it is expected that the PCI bus will be used for still some years. Consequently, we think our work is important.