Performance Analysis of the TCP/IP Protocol Under Unix Operating System for High Performance Computing and Communications[*]

Hyok Kim, Hongki Sung, and Hoonbock Lee
Hallym Information Technology & Electronics Research Center
Hallym University
Chunchon, Kangwon-do, 200-702, Republic of Korea
{hkim, hksung, hblee}@sun.hallym.ac.kr

Compilation : December 23, 2002

Abstract

Among many protocols, the TCP/IP protocol suite is the most widely used form of networking computers. With the advent of high speed communication paradigms such as asynchronous transfer mode, as well as with the advances of transmission medium technologies such as optical fibers, the physical transmission medium is no longer the performance bottleneck in communication systems. Thus, transport layers are currently receiving much attention since they don't seem to convey large bandwidths of lower layers to the application users properly. Many other transport protocols have been proposed and implemented, which might perform better than TCP/IP. However, due to the popularity of the TCP/IP, no other protocols seem to replace its place in the near future. Thus, emerging bandwidth hungry applications steadily require higher and higher performance of the TCP/IP. In this paper, we analyze the TCP/IP performance under UNIX operating systems in exchanging data once a connection is established. By measuring accurate data for the various aspects of the protocol implementation, we try to clearly illustrate the major bottlenecks and determine upper bounds on performance. We also measure memory bandwidth requirements in processing high-speed TCP/IP. Empirical results show that the TCP/IP protocol itself can handle up to 85 Mbps to process date under UNIX operating systems when proper data link layer interface is provided, requiring memory bandwidth of 172 Mbytes/sec.

1 Introduction

The TCP/IP protocol is truly the most widely used form of networking in today's computer communications. Since its first implementation in the late 1960s as a research project for packet switching networks, the TCP/IP protocol now forms basis for what is called the worldwide internet, a wide area network of more than forty million computers that literally spans the globe[11].

With the advent of high speed communication paradigms such Asynchronous Transfer Mode and

[*]This paper appeared in the *Proc. of High Performance Computing ASIA '97*, pp. 499–504, May 1997. This research was supported in part by a grant from the Ministry of Information and Communication of Korean Government.

optical fiber technologies, also with the demand for increase of emerging bandwidth-hungry applications, transport layers of communication protocols are receiving much attention to identify whether current transport protocols like TCP can adequately handle such high speed and high performance requirements. Limited bandwidth on the physical transmission media is no longer the performance bottleneck in communication systems. Instead, processing of communication protocols inside the network nodes is the most significant limiting performance factor of high-speed networks.

There exist two major approaches to achieving high performance transport components: (1) Design of new high-speed protocols [2, 3], and (2) Optimization of implementing existing protocols [4, 7]. Although several new protocols have been proved and demonstrated to give higher performance than TCP/IP, they still remain in the laboratories due to the popularity of the TCP/IP. It seems that no other protocols would easily replace the TCP/IP's place in the near future. On the other hand, several improvements of the TCP/IP have been proposed and made [4, 7]. Despite many arguments, it is still claimed that TCP/IP is capable of serving the high performance demands of future networks.

It has been known that major performance bottlenecks in TCP/IP processing are data copies, checksum calculation, memory management, timers, etc. Most of them lie in the interaction of the protocol with the operating system, not in the protocol itself. Several measurement techniques have been demonstrated to quantify bottlenecks [5, 8, 9]. Software based measurements can trigger operating system interrupts to acquire the current time which requires the execution of numerous additional instructions such as saving registers, memory swapping, etc. Thus, such techniques might give unacceptable perturbation. Hardware based measurement techniques are desirable for more accurate analysis. Specially designed hardware [10] or logic analyzer [8] have been used to measure accurate execution times. However, such methods have limited flexibility and cannot measure memory bandwidth requirements properly.

In this paper, we try to quantify the performance bottlenecks of the TCP/IP protocol running in the UNIX operating system. We focused the performance analysis on exchanging data once a connection is established. The purpose of our measurement and analysis is to see how much time is spent in the TCP and IP layers to send a block of data and to process the acknowledge returned by the receiving party. We also measured how many memory accesses is required and how many instructions are executed in doing so. In addition, we measured time spent in the UNIX operating system in sending data from user process to the physical network.

2 Measurement of TCP/IP performance under UNIX operating system

Current UNIX(4.4-BSD Lite) implementation of the TCP/IP protocol consists of four layers below the application: The socket layer, the TCP layer, the IP layer, and the data link layer. The socket layer gives the interface between the application and the TCP protocol, the TCP layer concerns with the end-to-end communication functions, the IP layer with the addressing and routing of packets, and the data link layer with getting the packet onto and off of the actual transmission media.

2.1 Methodology for Performance Measurement

We focus only on the performance, in terms of execution time and memory bandwidth requirements, of the data exchange once a connection is established, although overhead of connecting and disconnecting is also a performance concern. The purpose of this kind of measurement is to quantify performance bottlenecks in TCP/IP protocol processing in conjunction with UNIX operating systems. Since the time required for connection establishment and disconnecting it is much dependent on the communicating party's response, and also since it might not happen frequently in transmitting large volume of data such as multimedia data transmissions, we concentrate on the data exchange performance.

It has been identified by several researchers that major bottlenecks in TCP/IP processing under UNIX operating system are in data copy from user space to kernel space, another data copy from kernel space to the network device, checksum calculation, network buffer(mbuf) managements, etc. To quantify such bottlenecks, we set up the measurement environment as shown in figure 1 and 2. the parenthesized numbers in the figure 1 and 2 show the points where we measure the performance of TCP/IP processing. We use a data exchange scenario as follows. Suppose that a connection is established between the system under measurement and the communication party.

Figure 1: Performance measurement setup: sender's part. Two TCP/IP running computers are connected via an isolated Ethernet.

In figure 1, at(1), memory is allocated by the operating system to get a copy of user created data. Then, user data are copied from the user space to the kernel space(socket buffer) at(2). TCP checksum calculation is done at (3), and IP checksum at (4). Finally, data in the kernel space is transmitted to the network device at (5). Once data transmission is done, the system is waiting for an acknowledgment from the communicating party. When the acknowledgment arrives at the network device (6), IP processes it at (7), and then, TCP receives at (8). This concludes the data transmission.

In figure 2, at (1), a packet arrives in the network interface and the device driver enqueues the packet into the queue of IP. IP(2) and TCP(3) processing, such as address check, checksum calculation and connection name name check, is performed. Then the kernel copies data into the user buffer (4). If there is no error in the received data, an acknowledgment segment is constructed by TCP(5) and sent through IP (6) and the network device (7).

Figure 2: Performance measurement setup: receiver's part

For the system under measurement, we used a platform as follows: Our experimental machine was based on the Micronics M54Hi+ motherboard with a 150MHz Intel Pentium processor, a 256K byte cache, and 32M bytes of main memory. Our machine runs the FreeBSD which is a variant of UNIX and its TCP/IP implementation is based on the 4.4BSD-Lite. The Micronics M54Hi+ motherboard has a PCI bus and a ISA bus. The machine is equipped with a PCI Wide-SCSI controller and a 3Com ISA Ethernet controller.

2.2 Performance Measurement Tool and Metrics

We made our measurements using event/cycle counters implemented in the Intel Pentium processor. Although such counter are not documented for public users, they have been reported in [9] and an example application can be found in [1].

The Intel Pentium processor has one 64-bit cycle counter and two 64-bit software-configurable event counters. Each event counter can be configured to count one of a number of different hardware events such as data reads, data writes, instructions executed, hardware interrupts, data read cache misses, etc. In addition, the counters can be configured to count events in the kernel mode or in the user mode or in both. However, we counted events only in the kernel mode since the TCP/IP in the FreeBSD is implemented in the kernel. It should be noted that the cycle counter continues to increment when the machine is halted, but no other event counters are incremented.

As performance metrics, we measured cycle counts, instruction counts, data read counts, and data write counts. The cycle counts combined with the instruction counts will tell us the execution time and the complexity of the corresponding implementation. The data read/write counts will show us the memory bandwidth requirement of the corresponding function.

3 Empirical Results and performance Analysis

Using the measurement setup as described in figure 1 and 2, we measured execution time, instruction counts, and memory access counts when 100, 300, 500, and 1440 bytes of data are transmitted across the Ethernet. Figure 3 shows execution time(cycle counts) of socket, TCP, IP, and Ethernet when 100, 300, 500, and 1440 bytes of data are sent and then acknowledgment is received for the data sent. Figs. 4 and 5 depict instruction counts and memory access counts, respectively. Figure 6, 7, and 8 show the cycle counts, instruction counts, and memory access counts when receiving data across the Ethernet.

The graphs reveal that most time is spent in sending and receiving data inside the data link layer, Ethernet. This is mainly due to the relatively low speed(10Mbps) of the transmission medium. So, we disregard the latency generated by the data link layers to see the performance of the TCP/IP itself under the UNIX environment. From the graphs, time spent to send 1440 bytes of data in socket, TCP, and IP layers and then to receive an acknowledgment is approximately 0.21 *ms.* Time spent to receive 1440 bytes of data in IP, TCP, and socket layers and then to send an acknowledgment is about 0.19 *ms.* Thus, the TCP/IP under UNIX environment can process data up to 56Mbps for sending and 62 Mbps for receiving, requiring memory bandwidth of 61 Mbytes/sec and 172 Mbytes/sec, respectively.

If we further neglect the acknowledgment processing overhead, we might be able to see upper bounds of data sending and receiving performance of the TCP/IP under UNIX environment. The graphs show that time required to send 1440 bytes of data through socket, TCP, and IP layers without receiving acknowledgment is about 0.13 *ms.* Time required to receive 1440 bytes of data without sending acknowledgment is about 0.14 *ms.* Therefore, the TCP/IP under UNIX environment can send and receive data up to 87 Mbps and 85 Mbps when data acknowledgment is properly processed on bulk data transmission [6], in addition to proper data link interface is provided. This is an upper bound performance measured in our experimental setup.

4 Concluding Remarks

To analyze performance of the TCP/IP under UNIX operating system, we focused on the performance of the data exchange during a regular data transfer once a connection is established. Performance is measured in terms of cycle counts, executed instruction counts, data read counts, and data write counts. Since the cycle counter Intel Pentium processor continues to increment even when the machine is halted, also since it increments when the processor is switched to run other processes and interrupts, it is very hard to measure the precise execution time of the program block being executed. To ensure the measurement as accurate as possible, we repeated the same measure several time to take out such results which are far off the standard deviation.

We should note that the perturbation of measurement is not due to the measurement errors but due to the running conditions of the Pentium processor. For

Figure 3: Measurement of cycle counts in TCP/IP send processing

Figure 5: Measurement of memory access counts in TCP/IP send processing

Figure 4: Measurement of dynamic instruction counts in TCP/IP send processing

Figure 6: Measurement of cycle counts in TCP/IP receive processing

Figure 7: Measurement of dynamic instruction counts in TCP/IP receive processing

Figure 8: Measurement of memory access counts in TCP/IP receive processing

example, depending on instruction and data cache miss ratios, the cycle counts vary greatly. Also, the instruction counts are much affected by the internal pipelines state of the processor. Therefore, our measurement data are taken only when cache miss rates and pipleline state are within some specified ranges, discarding ones far out of ranges. By doing so, we can have an idea of relative execution time of the program under measure using the cycle counts in conjunction with the event counts such as instruction counts and data read/write counts.

We also did not consider functions which are not directly related with the regular data transfer such as urgent segment, error messages, etc. Thus, the overall performance which might be drawn from our empirical results should imply upper bounds on performance.

References

[1] J. B. Chen and et al. The Measured Performance of Personal Computer Operating Systems. *ACM Trans. Computer Systems*, Feb. 1996.

[2] D. Cheriton. VMTP as the transport layer for high-performance distributed systems. *IEEE commun.*, June 1989.

[3] G. Chesson. XTP-protocol engine VLSI for real-time LANs. *EFOC/LAN 88 (Amsterdam,Hollan)*, 1988.

[4] D. D. Clack and V. Jacobson. An analysis of TCP processing overhead. *IEEE Commun.*, 27:23–29, June 1989.

[5] J.-H. Huang and C.-W. Chen. On Performance Measurements of TCP/IP and its Device Driver. *IEEE Proc. of 17th Conference on Local Computer Networks*, pages 568–575, 1992.

[6] V. Jacobson. 4BSD TCP Header Prediction. *Computer Commun. Review*, 20(2):13–15, Apr. 1990.

[7] V. Jacobson. Efficient protocol implementations. *ACM SIGCOMM 90 Tutorial*, Sep. 1990.

[8] J. Kay and J. Pasquale. The Importance of Non-data Touching Processing Overheads in TCP/IP. *ACM SIGCOMM Computer Communication Review*, 23:259–268, Oct. 1993.

[9] T. Mathisen. Pentium secrets. *Byte*, pages 191–192, July 1994.

[10] A. Mink and et. al. Hardware Measurement Techniques for High-speed Networks. *Journal of High Speed Networks*, 3(2):187–207, 1994.

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