On Performance Measurements of TCP/IP and its Device Driver

Jau-Hsiung Huang and Chi-Wen Chen

Department of Computer Science and Information Engineering
National Taiwan University, Taipei, Taiwan, R.O.C.

Abstract

Among all protocols, TCP/IP is one of the most popular protocol suites in use. Hence, the performance behavior of TCP/IP becomes a very important issue and much work has been conducted in this area. In this paper, a performance measurement of the processing overhead of TCP/IP on personal computers interconnected by Ethernet is given. In this measurement, we found that most of the processing overhead comes from TCP and the Ethernet device driver. Among TCP, a large portion of overhead comes from the checksum computation. Further, almost all overhead of the lower layers comes from moving data from the main memory to the Ethernet card. Hence, if the bus speed can be increased and the TCP checksum can be performed by hardware, the processing overhead generated from TCP/IP and lower layers can be greatly reduced. The results presented in this paper shed light on designing communication protocols on personal computers.

1. Introduction

The motivation of this paper came from the industry with the following question. Should we implement TCP/IP [1,2] in silicon? In order to answer this question, we performed a series of measurement to find out the bottleneck of TCP/IP and its lower layers including the device driver. Although some results of the performance of TCP have been published [3,4], our measurements showed some results of our own by considering not only TCP but also IP and the lower layers.

Looking at a computer network, there are three main components, namely, computers and their operating systems, communication hardwares, and communication softwares. Among all combinations of these three components, an MS-DOS based personal computer network interconnected by Ethernet running TCP/IP is one of the most popular environments.

In [3], a performance study of TCP based upon UNIX on a Intel 80386 processor was conducted. In their work, they concluded that TCP is in fact not the source of the overhead often observed in packet processing, and that it could support very high speeds if properly implemented. However, they did not study the overhead generated from the checksum computation in TCP and the overhead from the lower layers and the device driver. In this work, a TCP source code was written for the performance evaluation.

In [4], a simple mathematical model was proposed to study the performance of TP-4 [5], which is in essence similar to TCP. This work studied the performance of various ways of implementation from both the sender side and the receiver side. However, the authors did not attempt to search for the performance bottleneck. Moreover, this work based their argument purely from a mathematical model, many practical implementation issues were totally not considered.

In our experiment, we actually measured the overhead generated by TCP, IP, and the device driver. To perform this measurement, we developed our own TCP/IP from a commercialized product with some improvements. One improvement we made was to rewrite the checksum computation of TCP using assembly language instead of C language. In this measurement, a detailed measurement of the overheads of the checksum computation and the acknowledgement processing was conducted. Various combinations of parameters, such as the length of a segment and the processor speed, are also tested in the experiment to show their impact on the performance. Moreover, the architectural characteristics of personal computers were also considered which helped us understand better the performance of the system.

2. System Configuration and Measurement Methodology

This research was sponsored by Computers and Communication Lab. of ITRI, Taiwan.

0742-1303/92 $3.00 © 1992 IEEE

The measurement environment of this project is to have two PCs interconnected by an Ethernet running TCP/IP protocol suite for communication. The PCs used in the system are 386PC with 80386-33 CPUs and 64 Kbytes cache memory in the configuration. However, a similar experiment is also performed on 286PCs. On each machine, 4096 bytes of buffer is preallocated for the socket. Each transaction contains 20 Mbytes of data in all measurements and we allow no other stations to send other data on the Ethernet while the measurement is under way. In order to avoid extra delay, an ACK is sent by the receiver immediately after a packet is received.

Five different segment sizes are used in the measurement to show the impact of segment sizes. These five segment sizes studied are 256, 512, 768, 1024, and 1280 bytes. The TCP/IP function blocks of sending/receiving *data* packets of the measurement are shown in Figure 1 and the function blocks of sending/receiving *ACKs* are shown in Figure 2. The functional description of each block follows.

* Write_Socket: The output routine of the application program which sends a stream of data to TCP.

* Read_Socket: The input routine of the application program which receives the data stream from TCP.

* TCP_Request: The interface between the application program and TCP which processes a TCP user request (e.g., attach, bind, connect, listen, send, receive, disconnect, ..., etc.).

* TCP_Output: This routine figures out what should be sent and should the data be fragmented. This routine also decides when to send the segment(s) to IP.

* TCP_Input: This routine reassembles segments, stores data in the socket buffer, processes the ACK and sends the ACK.

* IP_Output: This routine fragments segments into packets if necessary and routes them.

* IP_Input: This routine determines if the received packets have reached the ultimate destination and reassembles them before passing them to its TCP; or, these packets will be further forwarded if the ultimate destination has not been reached.

* Forward_Packet: This routine converts the internet address of the destination into an Ethernet address (ARP function) and sends it.

* Driver_Output: This block outputs a packet to the Ethernet.

* Driver_Input: This block is activated from an interrupt and inputs a packet from the Ethernet.

In this measurement, we divide the CPU costs into two categories. One is CPU cost *per packet* and the other is CPU cost *per byte*. We define the CPU cost *per packet* as the CPU time required for every packet. This CPU cost only depends on the number of packets sent and received and not depend on the total data size. For example, the overhead from the Ethernet card driver and the packet header processing for TCP, IP, and ARP are CPU cost per packet. Similarly, we define the CPU cost *per byte* as the CPU cost required for every byte of data sent. For example, TCP checksum computation, moving data within main memory, and moving data from main memory to the LAN card are CPU cost per byte. Hence, this cost depends on the length of all data bytes. In this experiment, we first measure the CPU cost per byte in transmission and then measure the CPU cost per packet sent.

Max Segment Size	256	512	768	1024	1280
Packet Number	80000	40000	26666	20000	16000
(A) write_socket	3.41	2.53	2.20	2.08	1.98
(B) tcp_output	21.47	16.26	14.51	13.68	13.07
(C) ip_input	6.12	3.06	2.05	1.52	1.29
(D) driver_out	26.25	22.64	21.30	20.72	20.39
(L) driver_input(Ack)	6.33	3.13	2.09	1.55	1.21
(M) ip_input(Ack)	2.64	1.32	0.88	0.66	0.47
(N) tcp_input(ack)	13.69	6.97	4.87	3.44	2.62
Tx. Time(Sender)	79.20	55.37	47.40	43.39	41.03
CPU Time(Receiver)	77.72	53.99	46.08	42.18	39.82

Table 1. Time spent in each function block of the sending machine.(Time unit: seconds)

In the measurement, we are able to measure the CPU time required by each function block as shown in Figures 1 and 2. For example, the CPU time of the function blocks of the sending computer is listed in Table 1. In Table 1, the computers used are IBM PC-386 running at 33Mhz with 64 Kbytes cache memory. As mentioned earlier, the data transferred in this measurement is 20 Mbytes with a socket buffer size of 4096 bytes. In Table 1, five different maximum segment sizes are measured. The data in this table will be later analyzed in the following section. Note that the sum of rows (A), (B), (C), (D), (L), (M), and (N) should equal the value of the Tx. Time of the sender. However, a small error may incur due to measurement error.

Clearly, as the maximum segment size gets larger, the number of packets required to convey 20 Mbytes of data decreases. Hence, CPU cost per packet gets smaller, which consequently reduces the transmission time for both sending and receiving. However, notice that the time taken by driver_out does not change significantly over all segment sizes since this overhead is generated per byte, which does not vary significantly with respect to the

segment size.

3. Performance Measurements

In this section, numerous measurement results are presented. Many conclusions can be drawn through these measurement results. In our first set of measurement, a maximum segment size of 1024 bytes is used to send a block data of 20 Mbytes between two 386PCs. We show the overheads (in percentage) generated by TCP, IP, device driver, and writing the socket. The results are shown in Figure 3, from which we notice that the CPU time consumed by the device driver takes up more than half of the overall time while TCP takes almost 40% of time. The total CPU time consumed is 43.39 seconds.

From this measurement, we note that the time taken by IP is only 5%. More carefully examining the overhead taken by the device driver, we found that more than 90% of the time consumed by the device driver comes from moving data from the main memory to the buffer on the Ethernet cards. Once the data is in the memory buffer on the Ethernet card, the co-processor on the Ethernet card takes charge and consumes little of the CPU time. This result indicates clearly that the *bus speed* of PCs is too slow for high speed networking and hence becomes the major bottleneck. Luckily, the EISA bus on personal computer developed recently has a much faster bus speed than the AT-BUS used in our measurement.

By further measuring the overhead consumed by the checksum computation, we have the results as shown in Figure 4. Notice that the TCP checksum computation takes up almost 25% of the overall time. Note especially that in our experiment, the checksum computation is written in Assembly language, which is much faster than a high level language such as C language. If considering only TCP and IP, the TCP checksum computation consumes half of the overall processing overhead of TCP/IP. Although in our measurement a file transfer is considered, which will result in a higher percentage of checksum computation overheads, this result still gives us an idea about the burden of the checksum computation. The above result suggests that if the TCP checksum computation can be implemented by hardware, the computational overhead of TCP/IP can be saved by 50% and the overall time can be saved by 25%.

However, there are two reasons which make checksum computation slow on PC architecture. First, the high-low byte sequence of an integer (16 bits) stored in PC's memory is different from the order defined by TCP/IP. Hence, a reordering of the sequence of each integer is required before checksum computation. Second, there is no 1's complement addition instructions on PCs. Therefore, we have to use 2's complement addition first and convert it to 1's complement addition. Since these situations may not occur in workstations, it suggests that if a workstation is used instead of PC, the percentage of checksum computation time may be reduced.

Figure 5 shows the CPU overhead created by sending and receiving the acknowledgement of each packet in the measurement. This figure shows that acknowledgement process consumes roughly 10% of the overall overhead, which is not negligible in high-speed network. Clearly, if an ACK is sent for every two packets like many commercial products do, then the overhead of acknowledgement packets will be halved. This will induce a 5% save in processing overhead.

For the above measurements, the maximum segment size is set at 1024 bytes. In the next measurement, a maximum segment size of 512 bytes is conducted with all other parameters unchanged. This result is shown in Figure 6. Figure 6 shows that the computational overhead of TCP and IP are slightly increased while the overhead of the device driver is slightly decreased. This is caused of the increase of the per packet computation time of TCP and IP; hence, the overhead percentage of the device driver is reduced. Nonetheless, the basic observations obtained above remain unchanged.

An important measurement result is shown in Figure 7 which has the same system configuration as in Figure 3 except that the PCs used here use Intel 80286 CPUs. Figure 7 shows that the overhead percentage taken by TCP and IP increases significantly over the device driver for slower computers. This is caused by two reasons. First, TCP and IP are written in software, hence more time will be consumed by using a slower computer. Second, the bottleneck of the device driver comes from the system bus which does not differ significantly between PC-286s and PC-386s compared to the speed difference between Intel 80286 and 80386 CPUs. Hence the processing time of the driver does not change dramatically with the CPUs.

By comparing Figures 3 and 7, we show that for systems with more powerful CPUs, the bus speed becomes a very important issue affecting the networking performance. This result shows that if the Ethernet chipsets can be directly mounted on the main board (motherboard) of the personal computer, the networking performance will be greatly enhanced since the bottleneck of bus speed is avoided.

Figure 8 shows the CPU time taken versus the maximum size of segments. This Figure actually comes from Table 1 shown earlier. Note from Table 1 that the transmission time with 512 bytes segment size is roughly 20% more than that with 1024 bytes segment size. Worse yet, the transmission time with 256 bytes segment size is roughly 40% more than that with 512 bytes segment size.

Moreover, from Figure 8, we note that the CPU time rises sharply for smaller segment sizes. This result quantitatively suggests us to avoid small segment sizes if possible. Figure 9 shows the CPU time versus the number of packets sent. This Figure shows that CPU time is linearly proportional to the number of packets sent which is not too surprising.

We define stream size as the number of data bytes passed to TCP from the application program in one send_primitive call. Figure 10 shows the transmission time versus different stream sizes with maximum segment size equals 1024 bytes. Two sets of curves are shown in this Figure, one is with PUSH flag set and the other is without. Note the sharp rise in transmission time for smaller stream sizes, especially when PUSH flag is set. This result advices us to avoid using a very small stream size if possible.

Figure 11 shows the CPU time for seven of the most important function blocks versus different segment sizes. Note that in Figure 11 that the TCP overhead generated by acknowledgement rises sharply when the segment size gets smaller and smaller. Hence, the percentage of CPU time taken by TCP protocol grows with respect to smaller segment sizes. This result clearly suggest that the overhead of acknowledgement packets cannot be ignored, especially for smaller segment sizes. Another conclusion shown in this Figure is that as the segment size decreases, the number of packets sent hence increases, the overhead percentage of driver decreases while the overhead of TCP increases. Moreover, the overhead percentage increase by acknowledgement processing is the most significant.

4. Observations and Remarks

From the measurements mentioned above, we have the following observations and remarks.

(1) Moving data from one place to another is very CPU time consuming. The worst of it is to move data from the main memory to the buffers on the Ethernet card though the system bus. Unfortunately, although the computational power of CPUs advances rapidly, the speed of the system bus is still the bottleneck of the performance. Hence, directly mounting the Ethernet chipsets on the motherboard is important as the CPU gets faster and high speed networking capability is required.

(2) The TCP checksum is a computation intensive process. We suggest that this checksum computation can be implemented through hardware instead of using software. Moreover, the TCP checksum computation is a 1's complement sum, which can easily be implemented by simple logic in hardware.

(3) If only TCP and IP are considered, the overhead generated by TCP far outweighs IP. This shows that it does not make sense by only implementing IP in hardware and leave TCP in software.

(4) Even if TCP/IP is implemented totally in hardware, the CPU overhead can be reduced by only a half if an Ethernet card is used since the data moving time from the system to the Ethernet card constitutes the other half of the CPU time.

Table 2 shows the time taken by the IP_output function block with various sized of routing table using linear searching algorithm. This result shows that the searching time in making routing decision may be significant. Hence, if a large network is considered, we should be careful in designing the algorithm in searching routes from the routing table. For example, in a file transfer application, since consecutive packets will be sent to the same destination; hence, we should always store the route of current packet in cache since this route is very likely to be the route for the next packet. By so doing, we do not have to search the routing table for most packets; hence, a significant amount of overhead can be saved.

Destination address order in routing table	IP_Output CPU time (C)
1	1.52
4	1.99
7	2.47
10	2.92

Table 2. IP_output processing time versus different routing table with linear searching.

5. Conclusions

As mentioned earlier, the motivation of this work is to find out the performance bottleneck of TCP/IP and its lower layers. At first, we planned to measure the workload percentage of each function block in TCP/IP and lower layers. However, as the above results has shown, the overhead of moving data either within the main memory or from main memory to the Ethernet card is the major bottleneck which actually depends on the operating system rather than on the communication protocols. Another bottleneck comes from the checksum computation and acknowledgement processing.

If the above bottleneck can be removed, the overheads of all other function blocks are not significant. We take Figure 4 as the example. If the new bus architecture can be five times faster (eg., EISA bus for PCs) and the TCP checksum computation can be implemented in silicon, then the processing time of TCP/IP and lower layers can

be reduced by as much as 61%. That is, the processing time is 2.56 times faster. From this fact, we believe the major bottlenecks have been identified; hence, we do not go on measure the overheads of other function blocks.

From the above discussions, we have the following conclusions. (1) The overhead generated by IP is far less than that of TCP. (2) The overhead generated by the Ethernet driver is more than that of TCP/IP. (3) In TCP, the checksum computation is a major bottleneck. (4) The overhead generated by moving data around in the memory or moving data from the memory to the Ethernet card is significant. (5) The operating systems and the bus speed are two major sources of bottleneck. (6) Implementing TCP/IP in silicon is not necessarily required. However, if the Ethernet chipsets can be mounted on the main board, the bottleneck of the bus speed may be greatly relieved.

References

[1] Information Science Institure. Transmission Control Protocol NIC-RFC 793. DDN Protocol Handbook, Vol. 2, Sept. 1981,pp. 2.179-2.198.

[2] Information Science Institure. DARPA Internet Program Protocol Specification NIC-RFC 791. DDN Protocol Handbook, Vol. 2, Sept. 1981,pp. 2.99-2.149.

[3] D.D. Clark, V. Jacob son, J. Romkey, and H. Salwen, "An Analysis of TCP Processing Overhead," *IEEE Communication Magazine*, June, 1989, pp.23-29.

[4] B.W. Meister, "A Performance Study of the ISO Transport Protocol," *IEEE Trans. Comput.*, Vol.40, No.3, March 1991, pp.253-262.

[5] OSI Transport Protocol Specification, Standard ISO-8073.

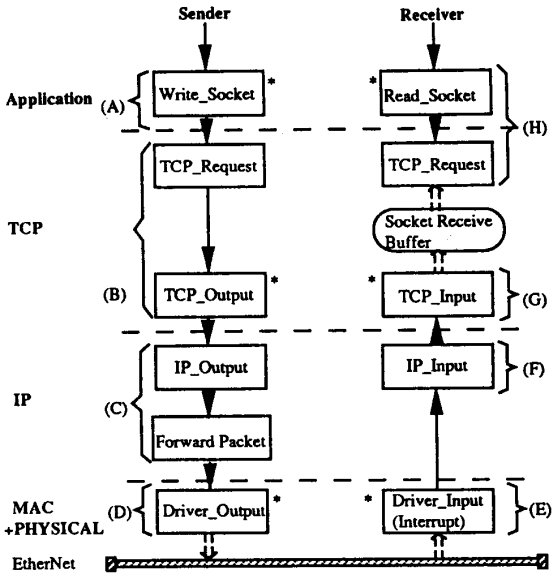

Figure1. TCP/IP Function Block Chart (Send a packet)

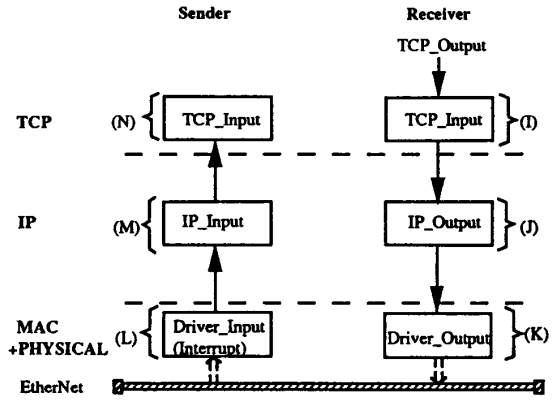

Figure2. TCP/IP Function Block Chart (Ack)

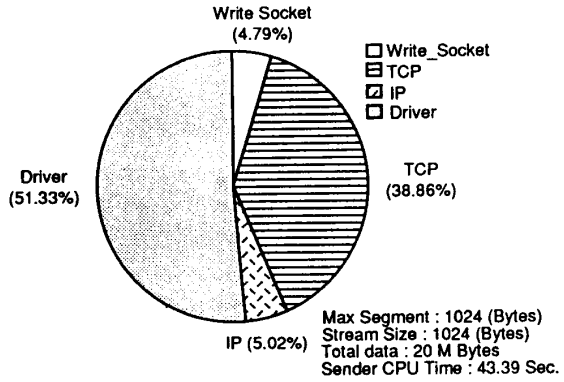

Figure 3. Ratio of CPU time with maximum segment size equals 1024 bytes

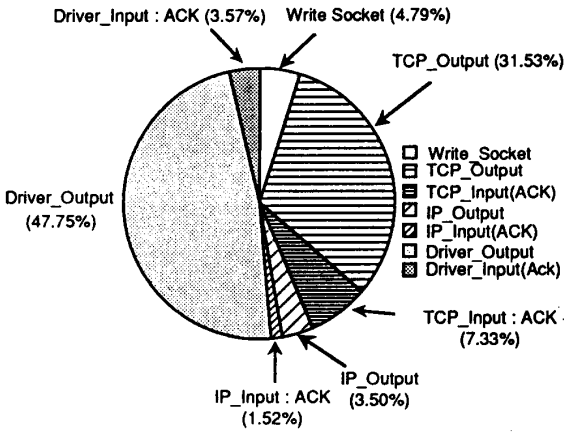

Figure 5. Ratio of CPU time with maximum segment size equals 1024 bytes

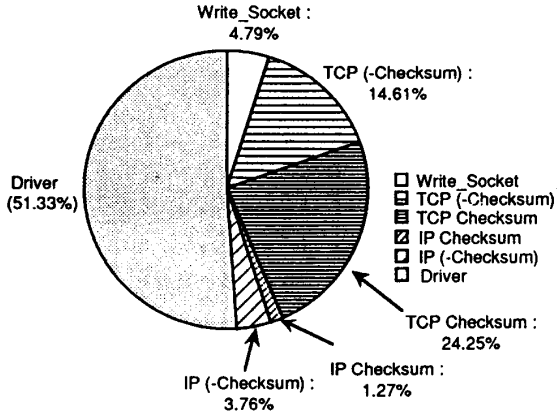

Figure 4. Ratio of CPU time with maximum segment size equals 1024 bytes

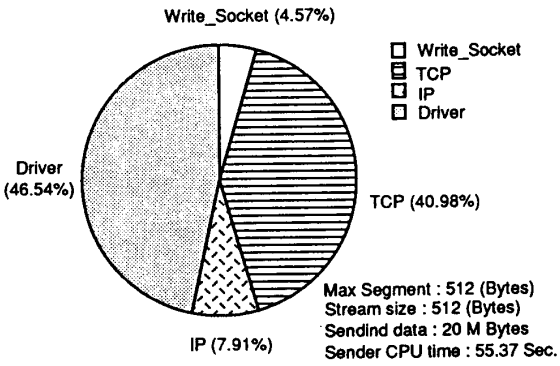

Figure 6. Ratio of CPU time with maximum segment size equals 1024 bytes

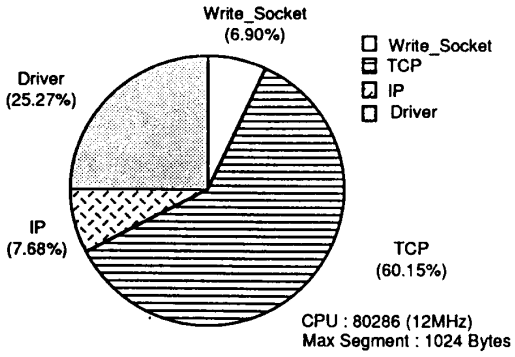

Figure 7. Ratio of CPU time with 80286 CPU

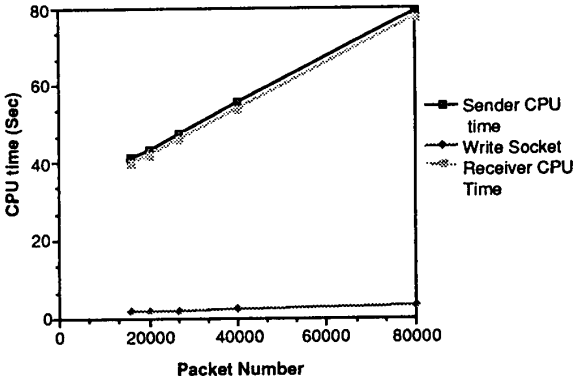

Figure 9. Sender CPU time versus the number of packet sent

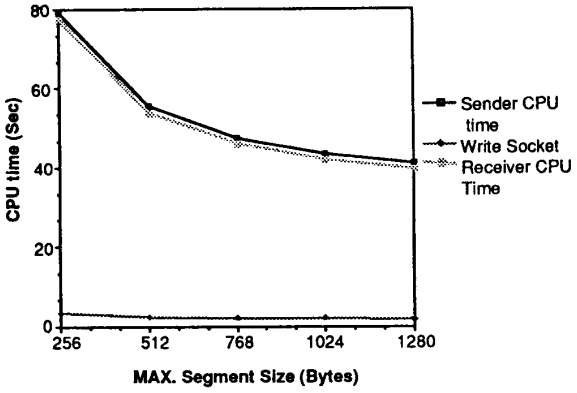

Figure 8. Sender CPU time versus different maximum sizes

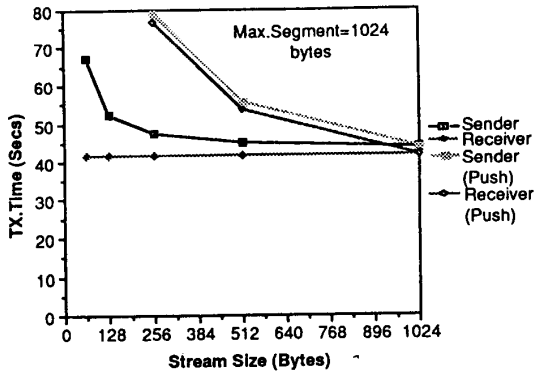

Figure 10. Sender CPU time versus different stream size (stream size=64,128, 256,512,1024 bytes)

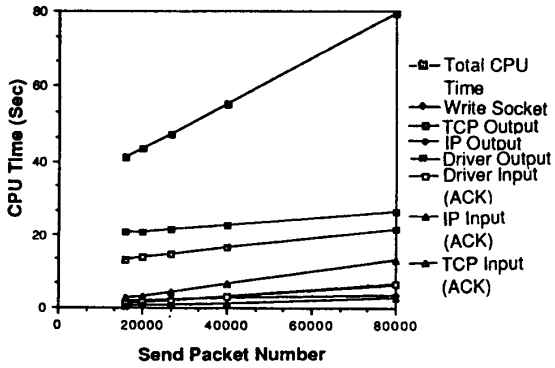

Figure 11. CPU time versus different maximum segment size