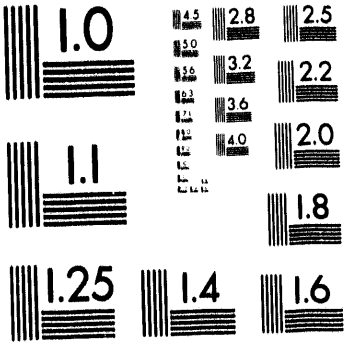

1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
6.3
7.1

1 of 1

UC-405

SAND 93-8243
Unlimited Release
Printed August 1993

TCP PERFORMANCE ANALYSIS FOR WIDE AREA NETWORKS

H. Y. Chen, J. A. Hutchins
Networking and Communications Department
Sandia National Laboratories/California
and
N. Testi
Networking Department
Sandia National Laboratories/New Mexico

Abstract

Even though networks have been getting faster, perceived throughput at the application level has not increased accordingly. In an attempt to identify many of the performance bottlenecks, we collected and analyzed data over a wide area network (WAN) at T3 (45 Mbps) bandwidth. The information gained will assist in designing new protocols and/or algorithms that are consistent with future high-speed requirements.

Contents

1. Introduction

While networks, wide area as well as local area, have been getting faster, perceived throughput at the application has not increased accordingly. One aspect of networking which is often suspected of contributing to low throughput is the Transmission Control Protocol (TCP) [1]. TCP is the transport protocol from the Internet protocol suite. It provides the functions of detecting and recovering lost or corrupted packets, flow control, and multiplexing. These functions are implemented using sequence numbers, cumulative acknowledgments, windows, and software checksums. This layer is typically executed in software by host processors, making it a source of processing overhead. This problem has been carefully analyzed by Van Jacobson et al [2]. Another TCP performance bottleneck is its poor utilization of paths with high bandwidth and long round-trip delay. The product of bandwidth and the round-trip time (RTT) is the number of bits it takes to fill the pipe (the physical link between two points), i.e., the amount of unacknowledged data that TCP must handle to keep the pipe full. The problems with TCP over such paths are window size limitation, costly retransmissions because of cumulative acknowledgment, and inaccurate RTT estimation. Work has been done with extensions to TCP to support reliable operation over high-speed paths, using window scaling, selective acknowledgment, and sender timestamp echo [3,4]. In this paper, we study these performance bottlenecks using an experimental testbed that simulates the conditions found in a wide area network, to understand better their impact and consequences. Data collected with the experimental testbed has been carefully analyzed, based on variables such as window size, link delay and maximum transmission unit (MTU). The information gained will assist in designing new algorithms for future high-speed networks.

The next section describes the data collection and analysis methodology. In Section 3 we analyze the observed statistical data and compare them with results from theoretical deductions. Section 4 summarizes the current work and outlines a distributed-application modeling project that will incorporate knowledge gained through this work as input for implementation and verification.

2. Testbed Configuration and Characterization

2.1 Testbed Configuration

Our experimental testbed (Figure 1) consisted of two FDDI rings, interconnected by two Network Systems Corp. (NSC) T3 routers. This testbed simulated a wide area network by using a T3 delay simulator developed at Sandia. There were two Sun workstations and one DECstation on each of the rings. The Sun workstations, with the exception of the SPARCstation 2, were dual-attach stations (DAS). The SPARCstation 2 and DECstations were single-attach stations (SAS) connected indirectly via a DEC FDDI concentrator 500. Two FDDI analyzers, one from TekeLec and the other from DTI, were used to collect trace reports for each experiment. We relied on these trace reports to study the delay characteristics of network devices and to verify network parameters (window size, packet size, etc.) employed in each test. Table 1 lists and describes all network components used in the testbed. Bulk data transfer rates were measured using the TTCP program, and we repeated each test three times for accuracy. TTCP is a program that measures the TCP memory-to-memory throughput between two machines. It provides the capability of performing these measurements with selectable parameters such as window size and the amount of data to send.

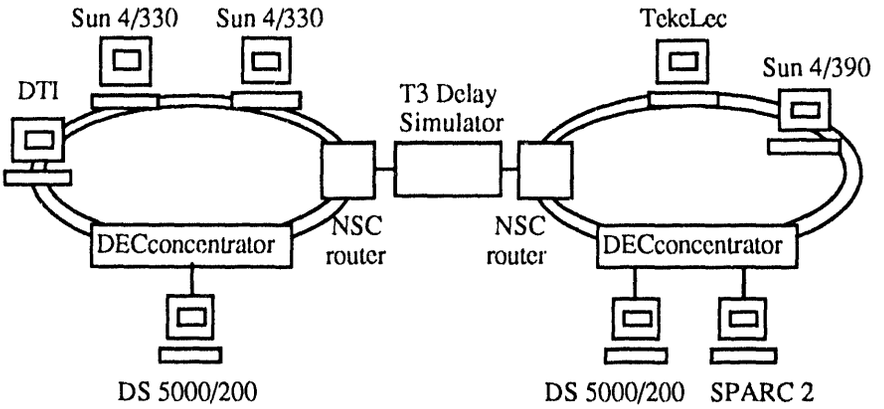


Figure 1. Experimental testbed topology.

Device	Description
DS 5000/200	16 MB, Ultrix 4.2, DEC FDDI TURBOchannel board ROM v1.0 Firmware v1.1, color monitor
Sun 4/390	32 MB, SunOS 4.1.1, SUN FDDI VME board Firmware 3.5, color monitor
Sun 4/330	8 MB, SunOS 4.1.1, SUN FDDI VME board Firmware 3.5, b&w monitor
SPARCstation 2GX	16 MB, SunOS 4.1.1, SUN SBus FDDI board (early release), color monitor
NSC Router	N703 with FDDI and T3 interfaces, Nucleus v 7.1, IP v2.3-10, 703 v1.9-560, FDDI v2.4-121
T3 Delay Simulator	0 to 80-ms adjustable time delay with burst error capability
DECconcentrator 500	a dual attached intelligent FDDI wiring concentrator supporting up to 8 single attached stations
DTI FDDI Analyzer	LANHAWK 5750 with passive tap v1.32B
TekeLec FDDI Analyzer	V 2.0

Table 1. Testbed hardware description.

2.2 Testbed Characterization

This section describes the preliminary work that we have done to study the hardware as well as the software components of the testbed.

Router Capacity: The router forwarding capacity was established previously by the Sandia Consolidation Project team. They ran the TTCP code individually for each of the four pairs of hosts and established each pair's maximum throughput. The same TTCP code was then run concurrently on all four pairs of hosts. Since the sum of the individual performances exceeded the concurrent aggregate throughput, the router's forwarding capacity was found to be 37.2 Mbps [5].

DECstation 5000/200 Performance: Based on the trace report collected using the TekeLec FDDI Analyzer, the average interpacket gap for a sending DS 5000 was 1.52 ms for large packets (MTU = 4352 bytes), and the processing time of an ACK packet was 2.25 ms. This value indicated that a sending DS 5000 can process a maximum of 658 large output packets per second during a TTCP session. For small packets (MTU = 552 bytes) the interpacket gap was 0.426 ms, giving a maximum of 2347 packets per second, and the processing time of an ACK packet was 1.27 ms.

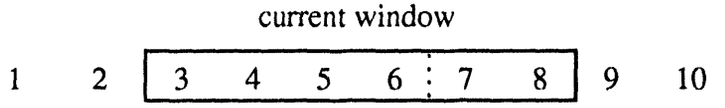
Sun 4 Performance: Based on the trace report collected using the TekeLec FDDI Analyzer, the average interpacket gap for a sending Sun 4 was 2.05 ms for large packets (MTU = 4352 bytes), and the processing time for an ACK was 1.40 ms. This value indicated that a sending Sun 4 can process a maximum of 488 large output packets per second during a TTCP session. For small packets (MTU = 552 bytes) the interpacket gap was 0.63 ms, giving a maximum of 1587 packets per second, and the processing time for an ACK was 0.56 ms.

T3 Delay Simulation: Propagation delay was introduced between the two remote NSC routers by adjusting a dial on the delay simulator. A baseline RTT of 3.1 ms between two

DECstations was established with the delay simulator switched off. This was the accumulated delays introduced by workstations and routers in the transmission path. Subsequent propagation delay values of 10 to 80 ms were calibrated accordingly using the baseline RTT of 3.1 ms.

TCP slow-start flow control algorithm: Trace reports of TTCP measurements are graphed to better understand the effect of the TCP slow-start flow control algorithm on the utilization of the communication bandwidth. Before we start our analysis of these trace reports, a glossary of terms is given to establish some basic TCP concepts [6].

> acknowledgment: a response sent by a receiver to indicate successful reception of information. It also advertises the amount of buffer space that the receiver has available for more incoming data.
>
> maximum transfer unit (MTU): the largest amount of data that can be transferred across a given physical network in a single packet. For local area networks like an ethernet, the MTU is determined by the network standard.
>
> maximum segment size (MSS): the largest amount of data that can be transmitted by TCP in one packet, it is usually calculated by TCP to equal to the MTU size minus the TCP and IP headers.
>
> window: the amount of data a sender is allowed to send before an acknowledgment arrives. The following figure depicts the TCP sliding window concept. Octets 1 and 2 have been sent and acknowledged, octets three through six have been sent but not acknowledged, octets 7 through 8 have not been sent but can be without delay, octets 9 and higher can not be sent until the window moves.



> congestion window: the congestion window is used by TCP to provide end-to-end flow control. At slow-start, its size will grow by one segment with each successful acknowledgment until it reached the maximum window size. When congestion occurs, the congestion window will decrease based on a back-off algorithm, with the minimum being one MSS.

a) A TTCP session between a DECstation 5000 on an ethernet and the Cray Y-MP/264 on the FDDI ring is shown in Figure 2a. The maximum TCP window size was set by TTCP at 32 KB, and the session used a maximum segment size of 1 KB. As shown by the graph, under the environment for a local area network, the slow-start algorithm set the initial TCP congestion window to the machine's default window size of 4 KB. Due to UNICOS's scheduling mechanism, the receiving application process was not immediately awakened to receive the incoming data. Normally, the receiving process reading the data causes TCP to send a window update. Since the receiving process was not reading the data, the connection was merely tagged as needing a delayed acknowledgment and one was sent at TCP's fast timer (200 ms) execution. The successful acknowledgment incremented the congestion window by one and again the sender sent a congestion windowful of packets. This pattern repeated until the maximum TCP window was reached. Meanwhile, there were 200-ms long idle times on the communication pipe. This is a typical case of a

bottleneck caused by the host processor, the Cray Y-MP's inadequate scheduling algorithm.

b) Figure 2b shows a TTCP session between two SPARC workstations with a maximum TCP window size of 8 segments. In this case, because the two SPARCS were remote to each other, the slow-start algorithm initialized the congestion window to one TCP segment size and incremented it by one segment with each successful acknowledgment delivery. Because the receiving SPARC was capable of keeping up with the incoming packets by processing them as they arrived, the machine followed the strategy of sending an acknowledgment when the available window had increased by two or more segments. Since the TCP window size was large enough to offset the RTT caused by link delay, we observed that the TCP flow-control mechanism allowed the communication path to fill after the slow-start. For this condition, the maximum throughput was governed by the available bandwidth.

c) Figure 2c presented a case where the long path delay was the performance bottleneck. This condition was introduced by limiting the maximum window size to the small value of four TCP segments. Again, the TCP window was initialized to one segment and incremented by one segment until it reached the maximum size. As shown by the graph, there were gaps in the pipe between some of the packets. The inefficient utilization of bandwidth caused poor throughput performance, which had gotten worse as the delay increased.

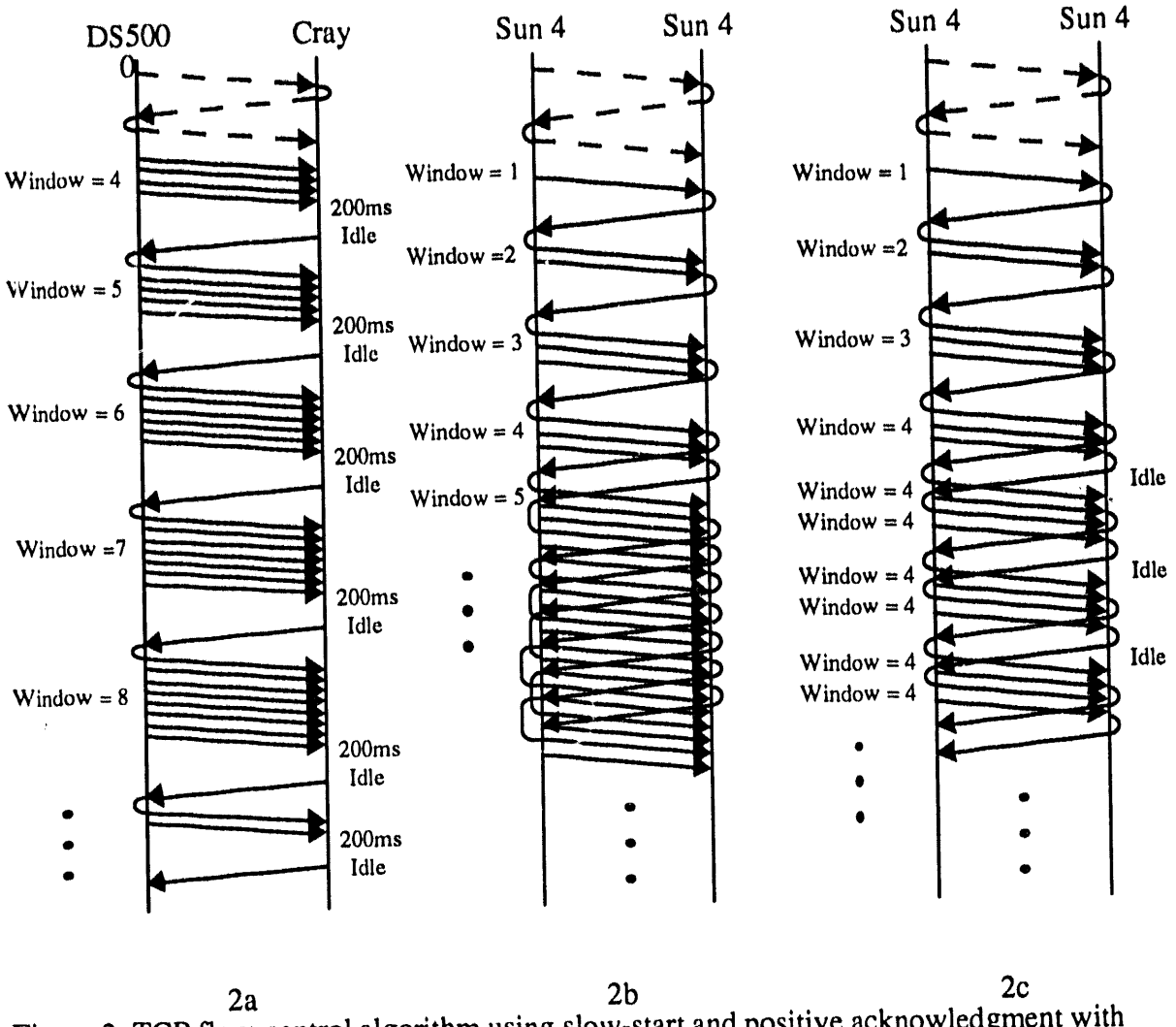


Figure 2. TCP flow-control algorithm using slow-start and positive acknowledgment with window advertisement. Vertical distance down the figure represents increasing time and diagonal lines across the middle represent network packet transmissions.

3. Performance Measurements and Analysis

We approached the performance of TCP for WAN conditions from two perspectives: 1) At a fixed link delay, we varied the maximum TCP window sizes and examined the shift of performance bottleneck from RTT to host processor, and 2) at a fixed maximum TCP window size, we varied the RTT of the link and examined the performance shift from host processor to RTT. Finally, the response time of request/response-based applications was studied against the link delay factors as well as the TCP window tuning.

3.1 Effect of TCP window size on throughput performance

The purpose of this experiment is to study the effect of TCP window sizes on the bulk data transfer rate. This test was carried out on a pair of Sun 4 workstations, once for local connections and once for remote connections at a round-trip delay of 42 ms.

3.1.1 TTCP performance of local FDDI workstations:

This experiment was carried out by running the TTCP code between two Sun FDDI workstations, a Sun 4/390 and a Sun 4/330, separated by a local NSC router. We conducted this experiment to demonstrate that when the bandwidth is high, even at 100 Mbps, the TCP window size can become a performance bottleneck in a local area network. Figure 3 illustrated the effect of TCP windows on TTCP performance of the pair of Sun 4 workstations. With an RTT of 3.1 ms and a bandwidth of 100 Mbps, the amount of data required to fill the pipe is 39 KByte (KB). As indicated by the graph, the throughput leveled out at a window size of 40 KB. The plateau of 13-14 Mbps represented the limitation of the Sun 4 workstation. Careful examination of the FDDI analyzer trace report revealed that the processing time of a large data packet on Sun 4 was 2.05 ms, the processing time of an ACK packet was 1.40 ms, and the average data-to-ack packet ratio was 2:1. Based on these statistics, the calculated maximum throughput of the Sun 4 was 12.77 Mbps, in close agreement to the observed value, 13.33 Mbps (Table 2).

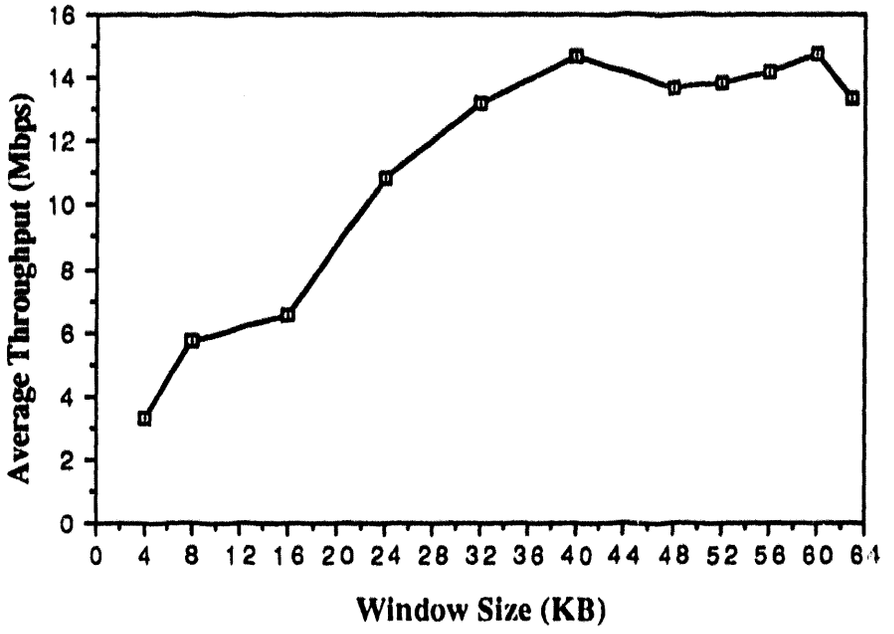


Figure 3. Effect of TCP window on throughput over local FDDI networks.

Window (KB)	Throughput #1 (Mbps)	Throughput #2 (Mbps)	Throughput #3 (Mbps)	Average (Mbps)
4	3.32	3.25	3.34	3.30
8	5.71	5.88	5.70	5.76
16	6.52	6.52	6.49	6.51
24	10.81	11.38	10.39	10.86
32	13.00	13.18	13.36	13.18
40	14.37	14.87	14.80	14.68
48	14.00	13.83	13.29	13.71
52	13.58	13.23	14.62	13.81
56	14.32	13.36	14.89	14.19
60	14.71	14.79	14.74	14.75
63	12.67	14.35	12.96	13.33

Table 2. TTCP measurements vs. varying TCP windows (FDDI SUN 4 workstations via local FDDI router).

3.1.2 TTCP performance of remote FDDI workstations:

In this experiment, the TTCP code was run between two Sun 4 FDDI workstations remotely connected via two NSC routers. Measurements were made by varying TCP window sizes at a fixed round-trip time of 42 ms. We explored the TCP performance bottleneck over a high-bandwidth and long-delay link. At 45 Mbps and 42-ms delay, the TCP window required to fill the transmission pipe is 236 KB. Anything less than 236 KB represented a potential performance bottleneck. This effect was shown in Figure 4. As the TCP window size increased, so did the throughput value. Theoretically, with a large enough window (236 KB), the throughput can equal the bandwidth, 45 Mbps. However, we believe that another performance bottleneck (caused by workstation's processing

power) will be encountered before reaching the T3 link speed. Indeed, previous experiments indicated that a Sun 4 workstation can send data at 13-14 Mbps over a local FDDI network, and this is the maximum performance that a Sun 4 can deliver.

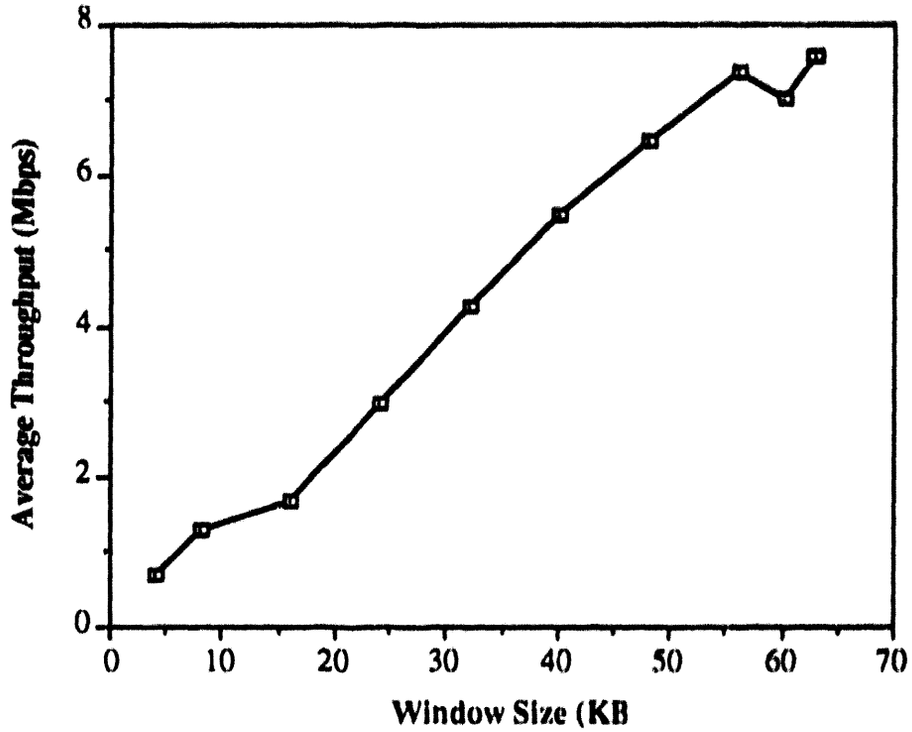


Figure 4. Effect of TCP window on throughput for remote FDDI stations.

Window Size (KB)	Throughput #1 (Mbps)	Throughput #2 (Mbps)	Throughput #3 (Mbps)	Average (Mbps)
4	0.66	0.66	0.66	0.66
8	1.29	1.29	1.28	1.29
16	1.66	1.69	1.69	1.68
24	2.97	2.94	2.95	2.95
32	4.25	4.24	4.29	4.26
40	5.48	5.57	5.38	5.48
48	6.20	6.72	6.40	6.44
56	7.50	7.25	7.30	7.35
60	6.92	7.65	7.82	7.80
63	7.23	6.38	7.69	7.10

Table 3. TTCP measurements versus varying TCP windows (remote FDDI SUN 4 workstations).

3.2 Effect of round-trip link delay on throughput

The purpose of this experiment is to show the effect of increasing link round-trip time on bulk data transfer rate. The TTCP code was executed on a pair of DECstation 5000/200's and the TCP window size was fixed at 63 KB. This test was repeated once with the workstation sending datagrams using the maximum transfer unit for FDDI network (4352 bytes), and once with the workstation sending datagrams using the default maximum transfer unit for remote networks (552 bytes).

3.2.1 TTCP performance using maximum transfer unit (4352 bytes)

The purpose of this experiment is to measure the effect of long link delay on throughput performance. Figure 4 showed the TTCP throughput values of a pair of DECstations across delay values ranging from 0 to 80 ms at 10-ms increments using a window size of 63 KB. The delay-bandwidth products were also listed for each delay (Table 4). It was clear that a TCP window size of 63 KB is sufficient to compensate for a delay value of 10 ms or less; beyond that the window size became the performance bottleneck. The bottleneck effect was demonstrated by the graph shown in Figure 5. As the delay-bandwidth product increased beyond the TCP window size of 63 KB, the DECstation experienced increased idle time waiting for an acknowledgment. This idle time resulted in a steady decline of performance for delays greater than 10 ms.

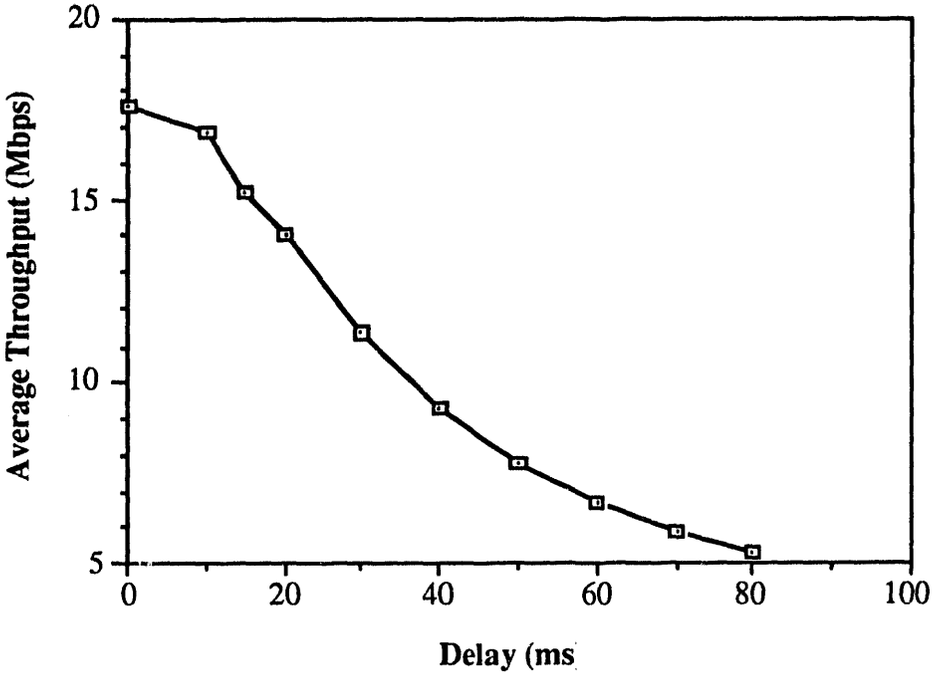


Figure 5. Effect of link delay on throughput (MTU = 4352 bytes).

Delay (ms)	Throughput #1	Throughput #2	Throughput #3	Average Throughput	Delay * Bandwidth (bytes)
0	17.75	17.40	17.63	17.59	NA
10	17.00	16.68	16.90	16.86	56,250
15	15.04	15.01	15.19	15.08	84,375
20	13.89	14.11	14.02	14.00	112,500
30	11.44	11.40	11.35	11.40	168,750
40	9.29	9.28	9.26	9.28	225,000
50	7.76	7.75	7.75	7.75	281,250
60	6.65	6.64	6.65	6.65	337,500
70	5.85	5.85	5.84	5.85	393,750
80	5.30	5.28	5.29	5.29	450,000

Table 4. TTCP measurements of throughput vs. link delay (MTU = 4352 bytes).

3.2.2 TTCP measurements using small packet size (552 bytes)

When an IP host sends data to a host on a different network, it uses a datagram size that is the lesser of 552 bytes or the first-hop MTU. This minimizes the probability of the need for fragmentation along the internetwork path. Since most of today's networks consist of ethernet (MTU = 1500) or FDDI (MTU = 4352), the datagram size selected was 552 bytes, which is much less than the maximum packet size allowed. This experiment measured the performance penalty incurred by using a small MTU. Table 5 summarized the TTCP measurements against varying delay values. By comparing results between Figures 5 and 6, we observed that 1) the maximum throughput had dropped from 14 Mbps to 7 Mbps, 2) the performance knee had shifted from 10-15 ms to 50-60 ms. Since using a smaller

packet size greatly decreased the efficiency of TCP processing by increasing the number of packets needed to send the same amount of data, it made the workstation become the performance bottleneck. The effect was a drop in performance and a shift of the performance knee to a larger delay value. We have demonstrated here the inefficiency of using a small MTU and thus the importance of an algorithm for dynamic discovery of path MTU [7].

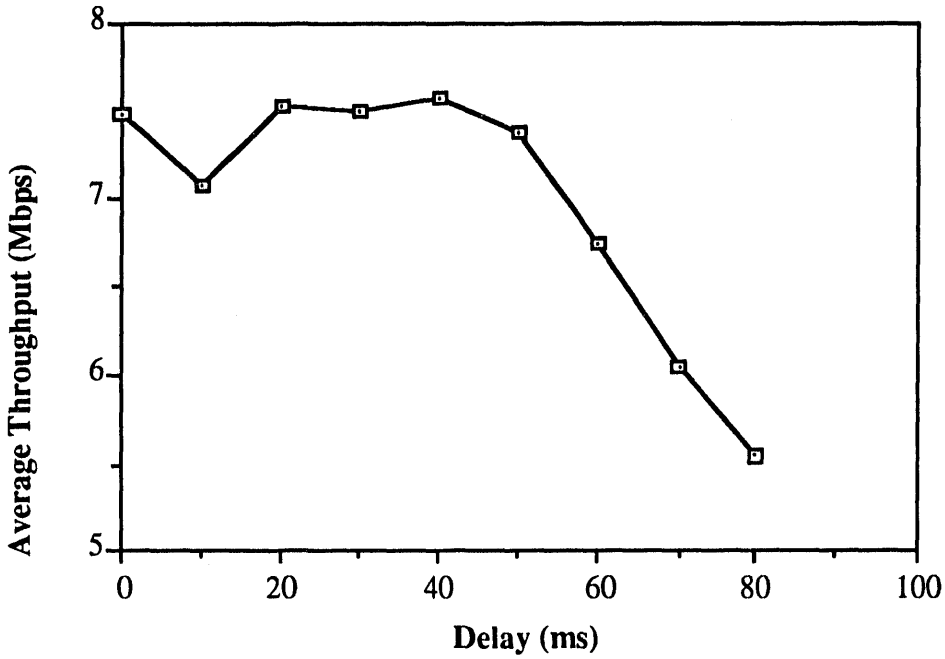


Figure 6. Effect of link delay on throughput performance (MTU = 552 bytes)

Delay (ms)	Throughput #1 (Mbps)	Throughput #2 (Mbps)	Throughput #3 (Mbps)	Average (Mbps)	Delay * Bandwidth (Bytes)
0	7.57	7.43	7.46	7.49	N/A
10	7.04	7.16	7.02	7.07	56,250
20	7.53	7.56	7.53	7.54	112,500
30	7.49	7.69	7.36	7.51	168,750
40	7.48	7.61	7.58	7.65	225,000
50	7.40	7.37	7.39	7.39	281,250
60	6.72	6.81	6.73	6.75	337,500
70	6.06	6.04	6.03	6.04	393,250
80	5.56	5.52	5.54	5.54	450,000

Table 5 TTCP measurements of throughput versus link delay (MTU = 552 bytes).

3.3 Effect of concurrent bulk data transfer to response time

Today's distributed computing environments consist of applications that are client/server based. They often use request/response techniques to achieve execution synchronization. Therefore, their efficiency is sensitive to increased RTT caused by link delay. The purpose of this experiment is to find out if the TCP window size has an additional effect on application response time when there is mixed traffic on the network. We generated mixload traffic by running one session of the request/response based echo program on the DECstations, and simultaneously, two sessions of TTCP code on Sun workstations. This experiment involved all six workstations on the testbed. During testing, the DECstations used a 63-KB window size, and the Sun workstations used a 48-KB window size. This experiment was conducted once with no round-trip link delay and once with a 42-ms round-trip link delay.

3.3.1 Mixload at 0 ms link delay

The response time and throughput metrics for the first mixload experiment were tabulated in Table 6. Figures 7-9 demonstrated the effect of TCP window size on application response time and on bulk transfer throughput rate. As shown by Figure 6, the increase in window size had a direct but small effect on the response time degradation. The degradation was probably caused by the competition at the router between the small application datagrams and the closely spaced bulk data transfer packets. Figures 7 and 8 also indicated that the TTCP throughput suffered from resource limitation on the router.

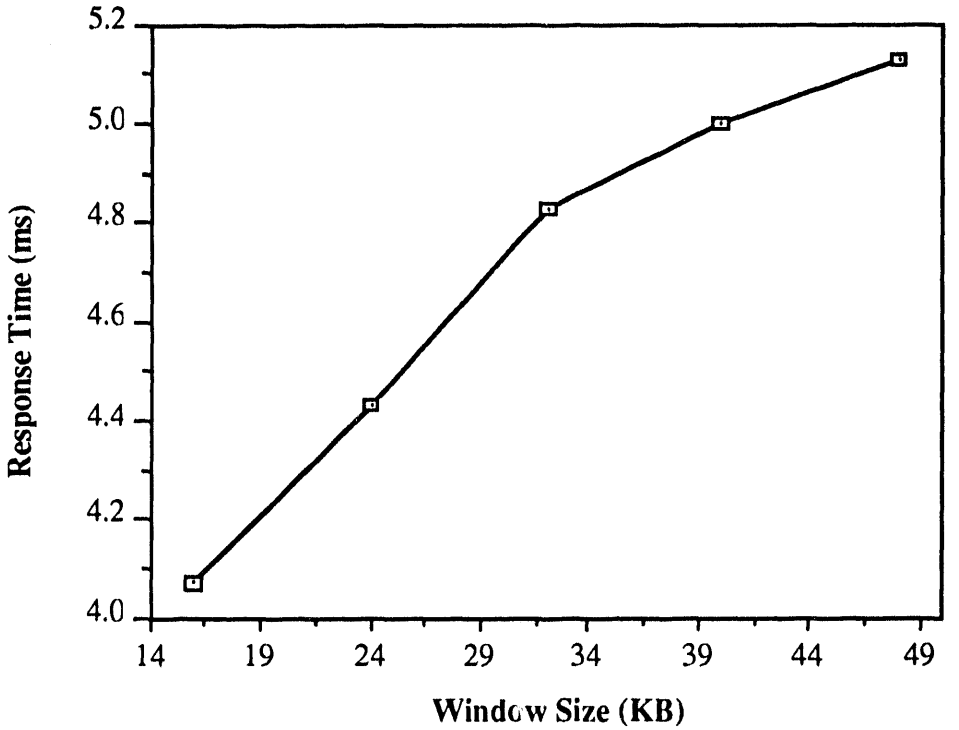


Figure 7. Effect of TCP window size on application response time (0-ms delay).

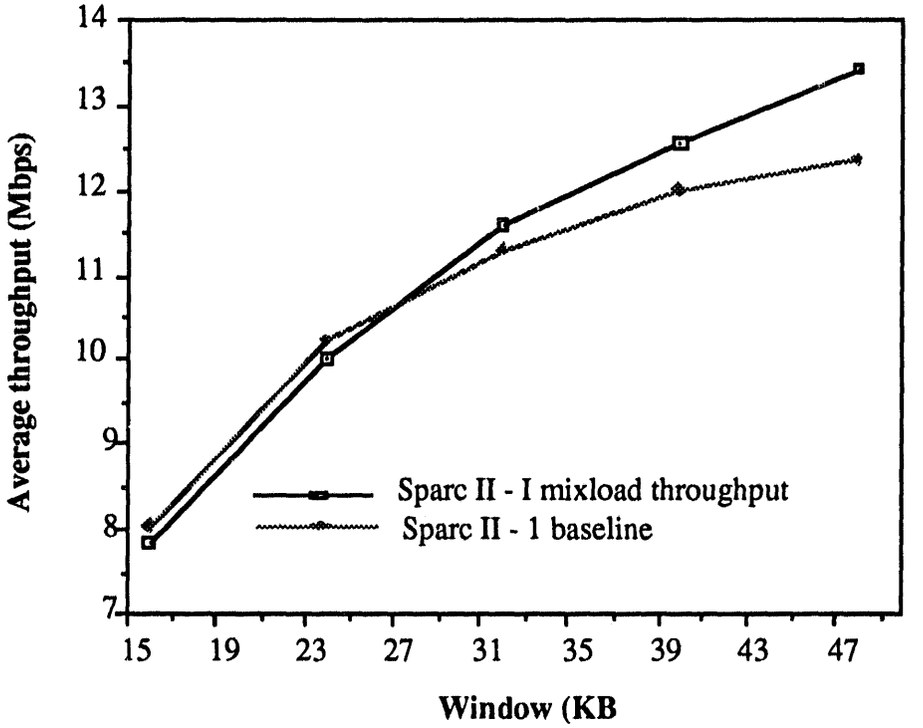


Figure 8. Effect of mixload traffic on bulk data transfer rate at 0-ms delay (SPARC 2-Sun 4).

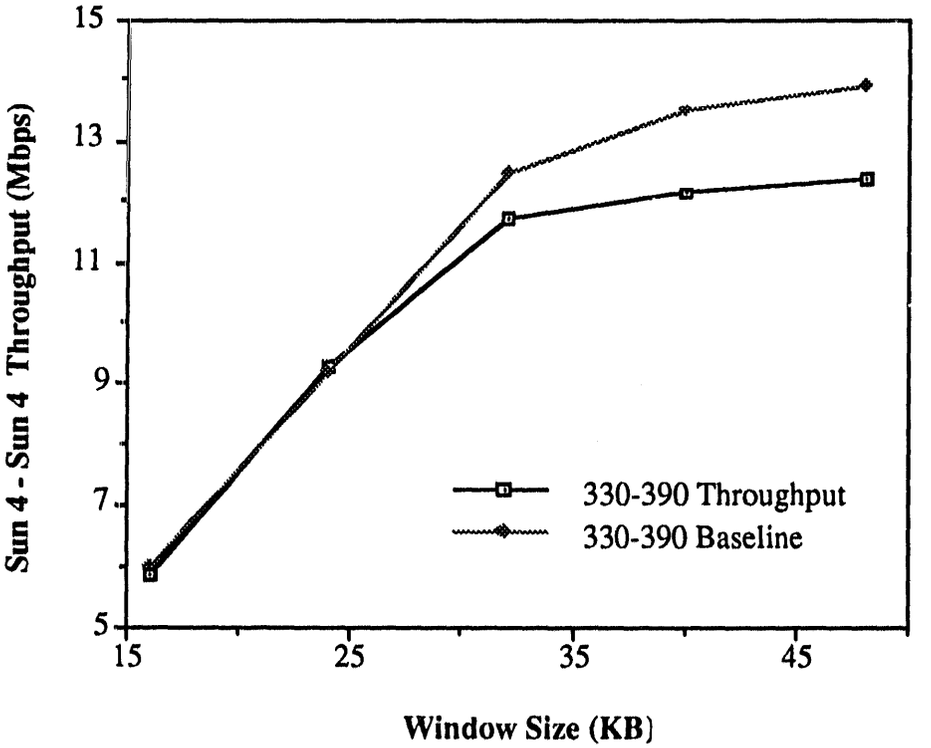


Figure 9. Effect of mixload on bulk data transfer rate at 0-ms delay (Sun 4 pair).

Window (KB)	DS 5000 pair Response (ms)	Sun 4 pair Throughput (Mbps)	Sun 4 pair Baseline (Mbps)	Sun 4-Sparc 2 Throughput (Mbps)	Sun 4-Sparc 2 Baseline (Mbps)
16	4.07	5.88	6.02	7.86	8.02
24	4.43	9.26	9.16	10.02	10.22
32	4.83	11.73	12.47	11.59	11.29
40	5.00	12.14	13.47	12.57	11.99
48	5.13	12.38	13.88	13.44	12.35

Table 6. Mixload concurrent TTCP measurements and echo response measurements (0 ms link delay).

3.3.2 Mixload at 42-ms link delay

The same mixload experiment was repeated by adding a 42-ms link delay between the two routers. Results (Table 7 and Figures 10) indicated that in this environment the TCP window size had even less impact on the application response time. The increased RTT caused by link delay actually provided a smoothing effect on the spacing between packets. We again observed a shifting of performance bottleneck from the router's forwarding capability to the long-delay path (Figures 11 and 12).

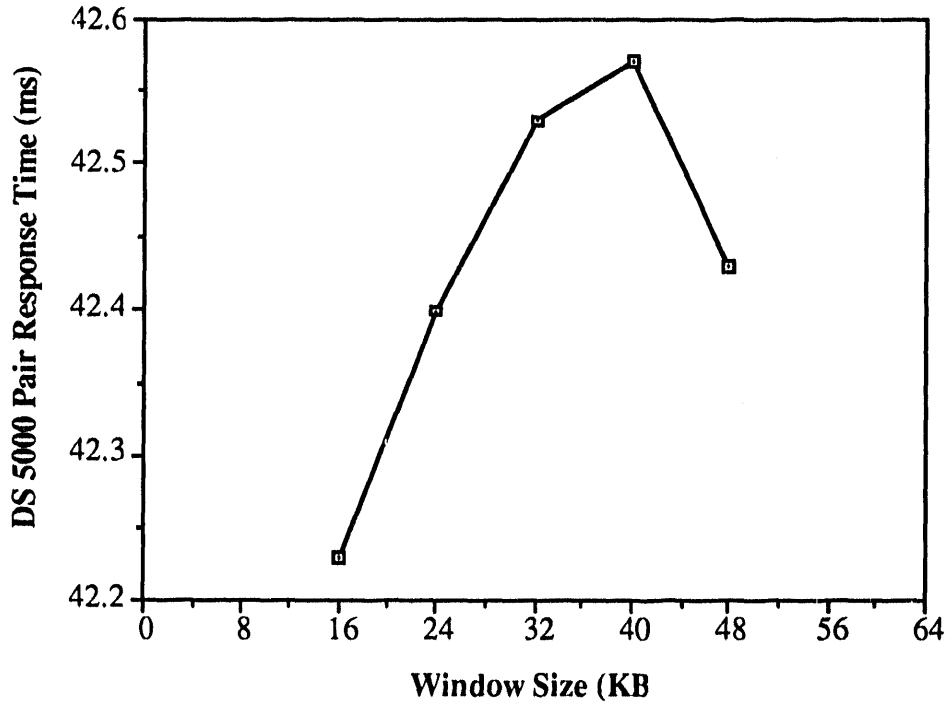


Figure 10. Effect of TCP window on application response time (42-ms delay).

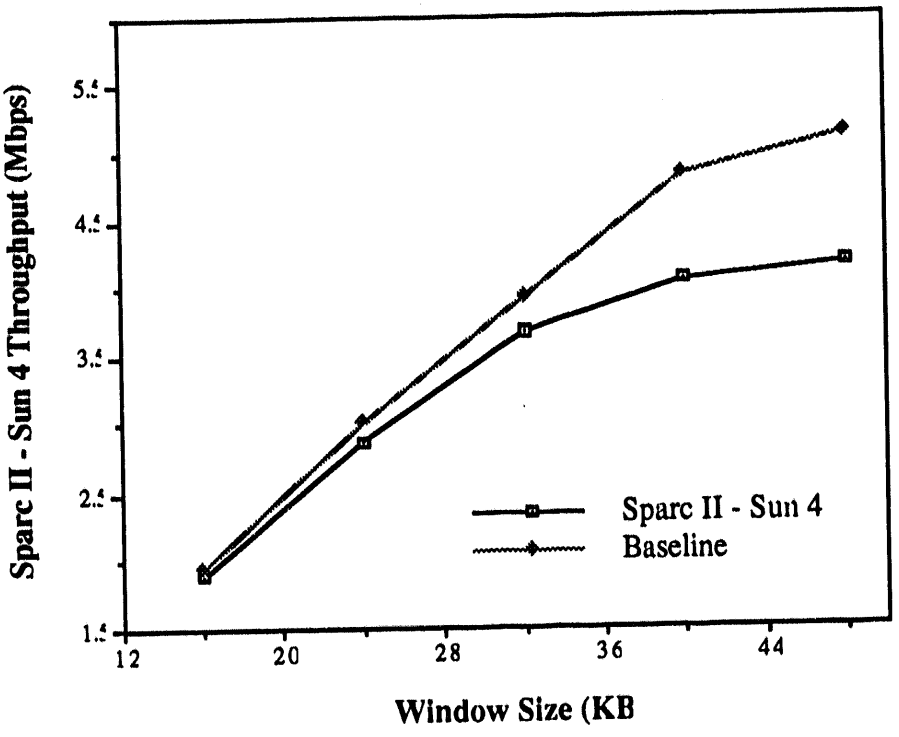


Figure 11 Effect of mixload on bulk data transfer rate at 42-ms delay (SPARC2-SPARC1).

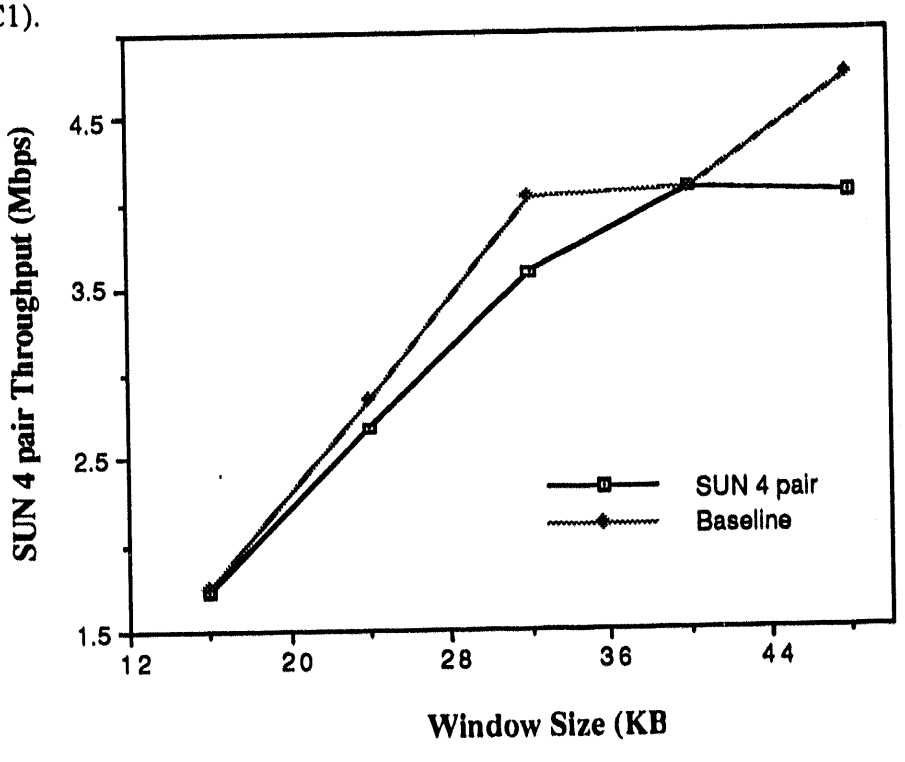


Figure 12. Effect of mixload on bulk data transfer rate at 42-ms delay (Sun 4 pair).

Window (KB)	DS 5000 pair Response (ms)	Sun 4 pair Throughput (Mbps)	Sun 4 pair Baseline (Mbps)	Sun 4-Sparc 2 Throughput (Mbps)	Sun 4-Sparc 2 Baseline (Mbps)
16	42.23	1.73	1.75	1.90	1.95
24	42.40	2.67	2.84	2.87	3.03
32	42.53	3.58	4.02	3.68	3.93
40	42.57	4.08	4.07	4.05	4.82
48	42.43	4.03	4.73	4.17	5.11

Table 7. Mixload concurrent TTCP measurements and echo response measurements (42 ms link delay).

4. Summary and Future work

In this study we have explored two fundamental performance bottlenecks faced by future networks, the protocol processing overhead on host processors and the difficulty of filling high-bandwidth/long-delay communication paths. We found that at T3 speeds and today's workstation power, TCP is sufficient for bulk data transfers at RTTs up to 10-15 ms. When the RTT is less than 15 ms, the TCP window is able to fill the communication bit pipe. However, when the RTT exceeds 15 ms, extensions such as window scaling, selective acknowledgment, and sender timestamp echo must be adopted. In addition, we studied the effect of maximum transfer unit (MTU) sizes on throughput performance. A small MTU requires the host computer to process a lot more TCP packets in order to send the same amount of data, and the resulting inefficiency degrades the throughput performance. We would like to see that the proposed dynamic path discovery algorithm, RFC 1191, be adopted by all TCP implementations. We have also examined the effect of RTT on distributed computing over wide area networks. Many distributed applications synchronize their execution using a request/response technique, and therefore the performance is sensitive to RTTs. Because the time spent waiting for a response to return over a long-delay network means more idle CPU cycles, the RTT effect on distributed applications will be more serious as workstation processors become faster. Since TCP does not address the RTT issue, we believe either a smart caching algorithm or a new lightweight protocol is needed to overcome this problem. Using a FDDI analyzer, we have measured various workstation and router delay characteristics and have studied the TCP slow-start algorithm. These data along with traffic pattern characterization results from previous works [8, 9, 10] will be the input parameters for a network performance modeling project. We hope that the output of the modeling project will provide us with an insight into the changes needed to adapt to future networks. It is also our plan to repeat this study when Asynchronous Transfer Mode (ATM) switches are added to our testbed. Because ATM technology will in all likelihood be the preferred choice for telecommunication vendors to provide Broadband ISDN services (voice, video, and data), it will impact wide area computing. We are interested in observing the effect, if any, of ATM technology on the data communications performance.

Acknowledgments

We would like to thank Tom Pratt for the use of the T3-delay simulator, John Naegle for his contribution in gathering network trace reports, and Bruce Whittet for building the testbed. Special thanks are also due to Rich Palmer, Thomas Jefferson, and Hilary Jones for their useful suggestions.

References

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

[2] D. D. Clark, V. Jacobson, J. Romkey, H. Salwen, "An Analysis of TCP Processing Overhead", IEEE Communications Magazine, pp. 23-29, June 1989.

[3] Information Science Institute. Transmission Control Protocol NIC-RFC 1072.

[4] Information Science Institute. Transmission Control Protocol NIC-RFC 1185.

[5] Nick Testi, Consolidation Project Report, March 1992.

[6] Douglas Comer, "Internetworking with TCP/IP", Prentice Hall, 1988.

[7] Information Science Institute. Path MTU Discovery NIC-RFC 1191.

[8] P. Danzig, S. Jamin, R. Caceres, D. Mitzel, D. Estrin, " An Empirical Workload Model for Driving Wide-area TCP/IP Network Simulations", Internetworking: Research and Experience, Vol. 3, 1-26 (1992).

[9] W. E. Leland, D. V. Wilson, "High Time-resolution Measurement and Analysis of LAN Traffic: Implications for LAN Interconnection", Proc. of INFOCOM '91.

[10] V. Paxson, "Measurements and Models of Wide-area TCP Conversations", Lawrence Berkeley Lab. TR LBL-30840, 1991.

UNLIMITED RELEASE

INITIAL DISTRIBUTION

1900	D. L. Crawford
1901	R. E. Palmer
1902	N. R. Morse
1903	D. C. Jones
1904	A. R. Iacoletti
1905	G. Gutierrez
1906	R. C. Dougherty
1932	C. D. Brown
1951	P. W. Dean
1951	D. E. Benthusen
1951	J. C. Berry
1951	F. T. Bielecki
1951	J. M. Brandt
1951	H. Y. Chen (10)
1951	C. Fang
1951	J. A. Hutchins
1951	Y. S. Yu
1951-1	D. H. Ching
1952	R. E. Cline
1954	M. O. Vahle
1954	A. Breckenbridge
1954	J. P. Brenkosh
1954	S. A. Gossage
1954	R. Hu
1954	J. H. Maestas
1954	J. H. Naegle
1954	L. G. Pierson
1954	T. J. Pratt
1954	N. Testi
1954	B. Whittet
1955-1	J. P. Sena
1956	R. J. Pryor
1957	W. D. Swartz
8000	J. C. Crawford
	Attn: E. E. Ives, 5200
	J. B. Wright, 5300
	M. E. John, 8100
	R. J. Detry, 8200
	W. J. McLean, 8300
	L. A. Hiles, 8400
	P. E. Brewer, 8500
	L. A. West, 8600
	R. C. Wayne, 8700
8535	Publications for OSTI (10)
8535	Publications/Technical Library Processes, 7141
7141	Technical Library Processes Department (3)
8523-2	Central Technical Files (3)

DATE
FILMED
11/5/93
END