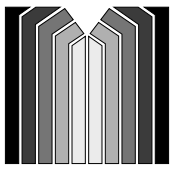
CITI-TR-89-3 June 12, 1989

On the Performance of Copying Large Files Across a Contention-Based Network

David W. Bachmann
Mark E. Segal
Toby J. Teorey

Center for Information Technology Integration (CITI)
The University of Michigan, Ann Arbor, MI 48105-2016

Center for Information
Technology Integration



The University of Michigan
Information Technology Division
2901 Hubbard
Ann Arbor, MI 48105-2016

On the Performance of Copying Large Files Across A Contention-Based Network

David W. Bachmann
Mark E. Segal
Toby J. Teorey

Center for Information Technology Integration
The University of Michigan
2901 Hubbard Avenue
Ann Arbor, MI 48105-2016

June 12, 1989

Abstract

Analytical and simulation models of interconnected local area networks, because of the large scale involved, are often constrained to represent only the most ideal of conditions for tractability sake. Consequently, many of the important causes of network delay are not accounted for. In this study, experimental evidence is presented to show how delay time in local area networks is significantly affected by hardware limitations in the connected workstations, software overhead, and network contention. The mechanism is a controlled experiment with two Vax workstations over an Ethernet. We investigate the network delays for large file transfers, taking into account the Vax workstation disk transfer limitations; generalized file transfer software such as NFS, FTP, and rcp; and the effect of contention on this simple network by the introduction of substantial workload from competing workstations. A comparison is made between the experimental data and a network modeling tool, and the limitations of the tool are explained. Insights from these experiments have increased our understanding of how more complex networks are likely to perform under heavy workloads.

1. INTRODUCTION

The large-scale interconnection of local area networks is an emerging area of intense activity and study, with applications covering academic, government, and business campus network configurations. Analytical and simulation models of such networks, because of the large scale involved, i.e. 10,000 nodes or more, are often constrained to represent only the most ideal of conditions for tractability. Consequently, many of the important causes of network delay cannot be accounted for and remain a mystery for

network managment. In this study, experiments are performed to show how delay time in local area networks is affected by hardware limitations in the connected workstations, software overhead, and network contention.

Researchers at the Center for Information Technology Integration (CITI) designed a series of network measurement experiments to test the accuracy of in-house developed analytical models of Ethernet and token ring networks, encompassed in a Network Modeling Tool (NMT). In conjunction with this work, the experiments attempt to determine how long it takes to copy a 100 MegaByte (MB) file between two computers on a local area network under a variety of network contention conditions and with several types of file transfer programs. To investigate these problems, a number of assumptions were made about the hardware and software used to perform the copy. Then experimental data was obtained from performing actual file copies between two computers at CITI, and the problem was modeled using the NMT. For this exercise, we were not interested in the "theoretical best performance" achievable with special-purpose, highly optimized copying programs, but rather how long a file transfer would take using standard tools under normal conditions and near saturation. We hypothesized that hardware speed limitations, software overhead, and network contention were all significant sources of file transfer time degradation from the ideal case, and were all within an order of magnitude of each other as causes of time delay.

File server performance was compared in an early study by Mitchell and Dion [1]. Diskless workstation activity was measured in [2], based on typical user activity patterns. This study was done using 4.2 BSD Unix, prior to NFS. It focused on congestion due to mutiple workstations, and evaluated different design and implementation alternatives using their model.

Measurement of network software delays was attempted by Bhargava, et. al. [3], who measured delay at the UDP level on a Sun and compared it to an experimental replacement they had designed. Measured delay for TCP and UDP in 4.2BSD was done in [4] , on both Sun II's and on Vax 11/780's and Vax 11/750's. They measured timings for thousands of packets on both unloaded and loaded ethernets, on both 10MBps and 3MBps Ethernet. They also used a profiler to breakdown time spent in the various routines that make up the BSD 4.2 TCP/IP and UDP/IP implementations. TCP overhead was found to be very small by [5]. FTAM and FTP protocols for file transfer, and their supporting protocols ROSE and SunRPC, were measured and compared in [6].

In Sec. 2 we describe the experimental methods used to obtain performance data. In Sec. 3 results of the experiments under contention-free network conditions are presented and analyzed based on our knowledge of the network testing environment. The experiments were repeated under several different network loads, including conditions very near saturation. The results from all experiments are then compared with predictions from

the Network Modeling Tool in Sec. 4, and conclusions we reached are summarized in Sec. 5.

2. EXPERIMENTAL PROCEDURE FOR NETWORK PERFORMANCE

To measure the time required to copy large files, two of CITI's Digital Equipment Corporation VAX computers were used with sufficient disk space to conduct the experiments. The configuration is shown in Fig. 1.

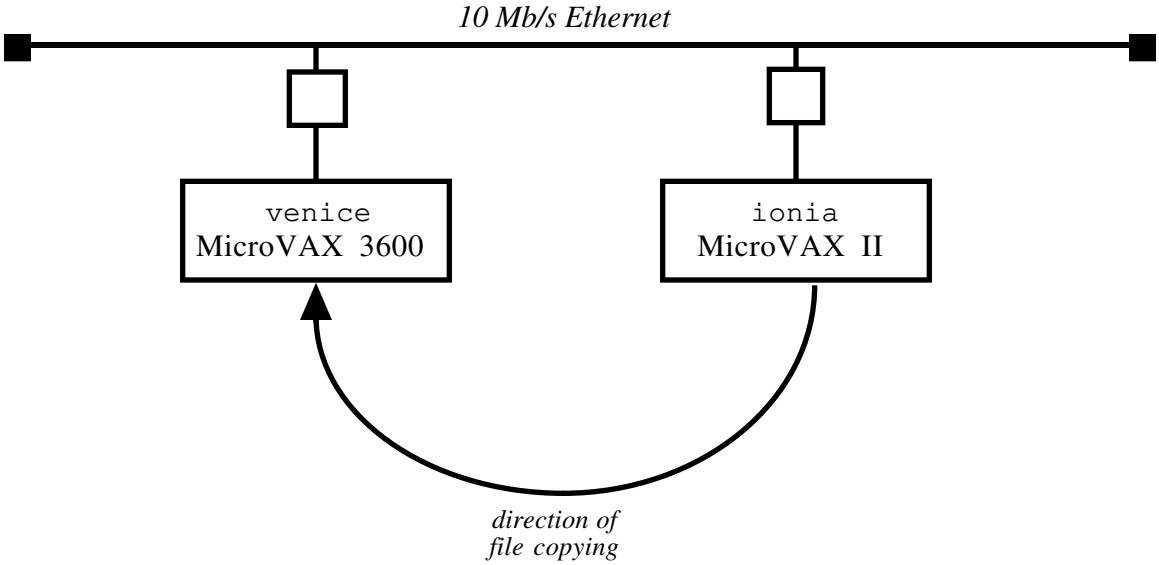


Figure 1 — Basic Test Configuration

The machine labelled `venice` is a MicroVAX 3600 CPU (approx. 2.5 MIPS) and the machine labelled `ionia` is a MicroVAX II CPU (approx. 1 MIPS). `Venice` has four RA82 disks connected to a single controller on the Q-Bus backplane while `ionia` has a single RA82 connected to a controller on its Q-Bus backplane. Both machines are running stock Ultrix 3.0 kernels. Ultrix is compatible with Berkeley Unix 4.3 BSD. Each CPU has 16MB of main memory, of which 1.6 MB is allocated for kernel buffers. The computers are connected together via thin-LAN Ethernet and were isolated from the rest of the CITI network for the duration of the workload-free experiments. The computers were brought up single-user and the necessary daemons (to support networking and NFS) were manually started from a single-user root shell. Since `venice` is faster than `ionia`, all machine-to-machine file copies were done from `ionia` to `venice`.

For the normal network load experiments, both machines were placed on the CITI Ethernet in their usual configurations: multi-user fileservers. The CITI Ethernet contains over 20 Unix workstations used for software development as well as a number of Macintosh computers connected to the Ethernet via a Kinetics FastPath Gateway. CITI has a Sun 3/280 fileserver

which is used primarily for software development and file service, while `venice` and `ionia` are also used as fileservers, but not for software development. CITI is connected to the rest of the University of Michigan by a Proteon gateway which is also attached to the CITI Ethernet. The normal load was augmented using the techniques discussed in Sec. 3.

To generate files to copy, a small C program was created to write files in multiples of 1 Megabyte. Test files of 1, 2, 5, 10, 25, 50, 75, and 100 MB were written for the experiments. These files were copied from `ionia` to `venice` using standard file copying programs available on most Unix systems that support networking. These programs are described below.

FTP `FTP` [7] is a program designed for large file transfers between local and distant remote hosts. The user typically logs into the remote computer with a valid login and password. Once this is done, files may be copied between the machines by giving `FTP` a number of `put` and `get` commands. In all cases, `venice` was logged into from `ionia` as `root` and `put` files from `ionia` to `venice`.

rcp `rcp` is an extension of the Unix `cp` (file copy) command that works across physical computers. The syntax is the same as a normal `cp` command except a file name may be preceeded by a host name and a colon. `rcp` will check for valid access permissions on the remote machine and then perform the copy. The user does not log into the remote machine explicitly; access must be pre-arranged (in a `.rhosts` file) before the copy is started. `rcp` is usually used for copying files between computers on the same local-area network, although it can also be used for file copying with distant hosts.

NFS NFS is Sun Microsystems' Network File System [8], [9]. NFS provides the capability for remote file systems to be accessible from normal Unix commands on the local machine. For example, the functionality of the rcp command described above can be duplicated in an NFS environment with the standard Unix cp command. Unlike FTP and rcp, however, NFS communicates between machines via a stateless (and connectionless) transport. NFS is built on top of XDR, RPC, UDP and IP protocols [10][11][12][13] while FPT and rcp are built on top of TCP and IP protocols [14]. Because NFS is a stateless filesystem, context must be established and access rights must be checked before every filesystem operation.

Files of the sizes described earlier were copied from `ionia` to `venice` using each of these methodologies. Each experiment was repeated several times to ensure accurate timing under Unix. Except for the heaviest background network loads, there was very little variation between the runs.

Baseline experiments were also run to determine how quickly a file could be copied on the same machine. Files were copied to the same disk of both machines. On `venice`, files were copied from one disk drive to another disk drive to evaluate bottlenecks due to disk scheduling.

3. EXPERIMENTAL RESULTS ON ETHERNET

3.1. Contention-Free Ethernet

The results of our experiments on a contention-free Ethernet are shown in Fig. 2. Each type of file transfer is indicated by the markings on its line on the graph. The ideal case, i.e. no network delays except for transmission delays, is also plotted.

Contributing factors. As can be seen from the graph, NFS is the slowest in all cases. This can be attributed to the inherent statelessness of the NFS protocol, which requires that context be established for each request, and to the nature of the RPC protocol over which NFS runs, which implies that each read or write request must complete before the next can be transmitted across the network.

Also, as expected, the `venice` disk copies are faster than all intermachine file transfers. The `ionia` to `ionia` file copy is slightly slower than the `rcp` but we believe this is due to the fact that this file copy was done to different disk partitions on the same physical disk (thus causing extensive disk head movement). The `venice` to `venice` different disk copy (dd on graph) was faster than all other methods of file transfer.

3.2. Ethernet Under Normal to Heavy Network Load

After running the experiments shown above, `venice` and `ionia` were re-attached to the CITI Ethernet and the experiments were repeated during normal working hours. Under this normal network workload, we discovered that the file transfer times using all three file transfer methods were nearly identical to the times observed on a contention-free Ethernet. As in the contention-free experiments, there was almost no variation between file transfer times on the normally-loaded network.

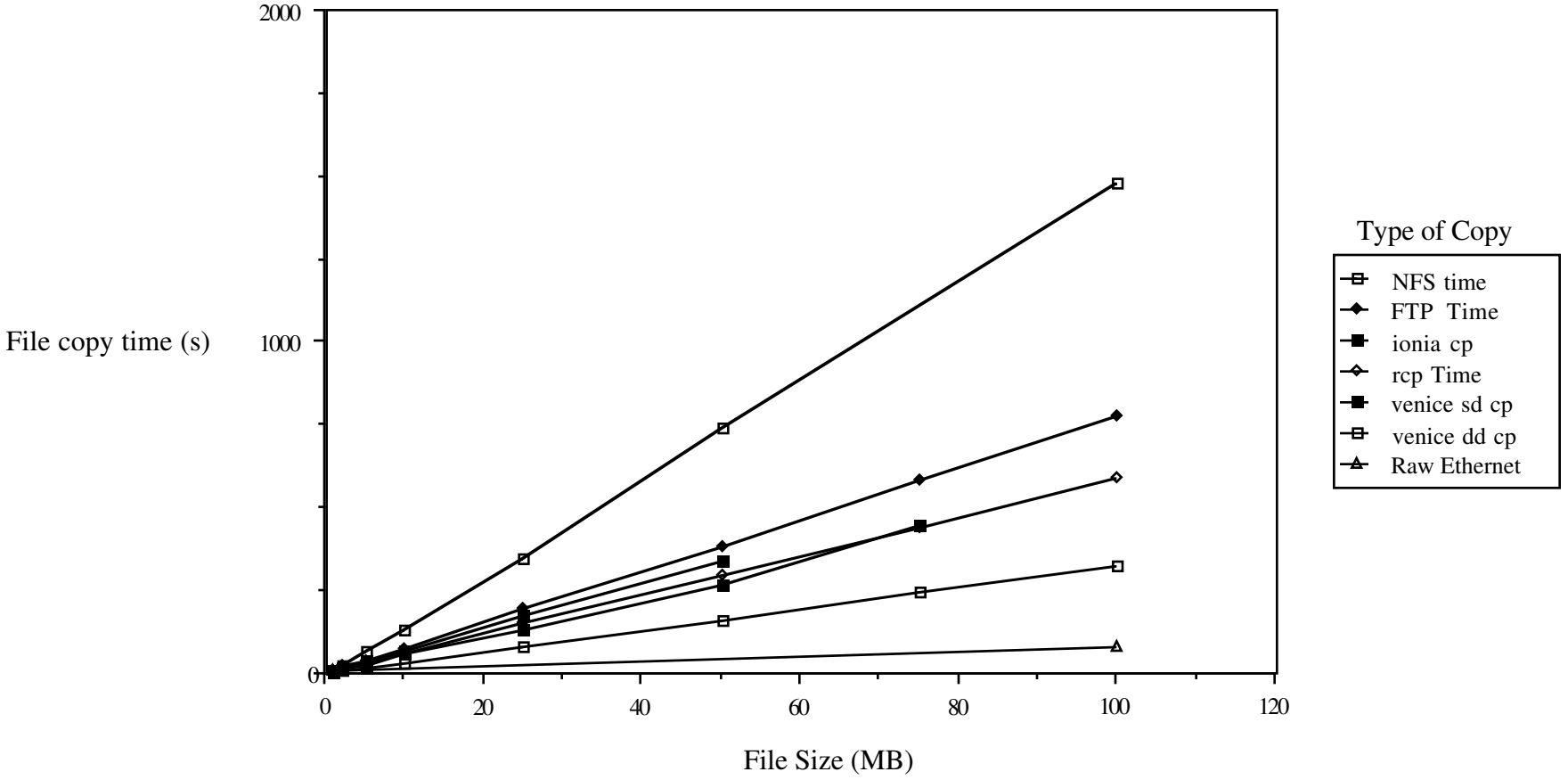


Figure 2 — Unix File Copy Performance on a Contention-Free Ethernet

These results were somewhat surprising since we believed our network was more heavily loaded than these experiments indicated. To investigate this further, a device called a "Sniffer", sold by the Network General Corp., was used. The Sniffer is an 80386-based IBM PC-compatible equipped with special hardware and software to monitor Ethernet traffic. It can be configured to monitor Ethernet utilization. The Sniffer was attached to the CITI Ethernet and discovered that the "normal" background load conditions present when these experiments were run never used more than 10% of the network capacity. Moreover, the network was less than 1% utilized much of the time. As a result, this experiment merely demonstrated that minimal network loads do not affect file transfer times significantly.

In an attempt to increase the Ethernet load to affect file transfers, a pair of programs written by a colleague at CITI that transmit and receive large numbers of UDP/IP packets on an Ethernet were obtained. One program performs the packet transmission (`udpsend`) while the other (`udprecv`) receives the packets. The receiver is used to count how many packets were successfully received, and to prevent error messages from being sent back to the sender (due to no process listening for its packets) which would then distort the packet size distribution on the network. `udpsend` was set up on machine `citi`, a Sun 3/280 fileserver (~4 MIPS), and `udprecv was set up` on `emerald`, a ~3 MIPS NeXT computer. When these programs are running, the network utilization (as measured by the Sniffer) is between 60% and 65% of its capacity. After these programs were started, the file transfer experiments were repeated. As in a lightly loaded network, the file transfer times were not affected by the background network load. Although this surprised us, it will be seen in Sec. 4 that this is in accordance with the predictions of our analytic models.

File copy time degrades as a function of increased network utilization, which is due to both an increase in the number of workstations at constant workload, or a constant number of workstations, each with increased workload. The relative effects of these two cases are currently under investigation.

3.3. Ethernet under Nearly Saturated Network Load

To obtain a quantitative measure of the degradation of file copy time under a nearly saturated network load, two pairs of `udpsend`/`udprecv` programs were set up since CITI had no pair of machines that can completely saturate the Ethernet. This configuration is shown in Fig. 3.

The experiments described above were repeated with two additional Sun workstations, `hbomb` (1.5 MIPS Sun 3/50) and `tahoe` (1.5 MIPS Sun 3/75) as shown in the figure. With these machines and `citi` and `emerald` exchanging UDP packets, the network utilization reached 98%. The file transfer experiment was then repeated for 10 and 20 MB files. The results of these experiments are plotted in Fig. 4. For comparision, the copy times of

the same files on a contention-free network are also plotted on the graph. As can be seen from the graph, the file copy performance under network near saturation conditions (the top three lines on the graph) is considerably slower than the contention-free file copy performance.

This experimental data indicates that hardware bandwidth limitations, software (file transfer) overhead, and network contention have similar order of magnitude effects on a 100 MB file transfer time. In Fig. 2 we saw that the hardware bandwidth limitations of the Vax machines increased file transfer time from 4-to-1 to 8-to-l over raw Ethernet speed. File transfer programs ran from 600 to 1500 seconds, for a total degradation of between 8-to-1 and 19-to-1 over the raw Ethernet transfer time of 80 seconds. In Fig. 4, when we extrapolate the times to 100 MB, the degradation due to heavy loading (near saturation) resulted in an increase of from 34-to-1 to 63-to-1 from raw Ethernet speed without contention. Taken individually, the effects on file transfer times were:

1. Hardware limitations: from 4-to-1 up to 8-to-1 longer transfer time.

2. Software overhead: from 1-to-1 up to 2.5-to-1 longer transfer time (depending on the file transfer program).

3. Network contention: average of 4-to-1 longer transfer time at 98% utilization.

In this set of experiments, there was significant variation between the times required to complete the individual runs of the file copy operations. This is due to Ethernet's design. Ethernet waits a pseudo-random amount of time before retransmitting a packet after a collision. Because many collisions occurred on the Ethernet during these experiments, the exact behavior will be different for each run.

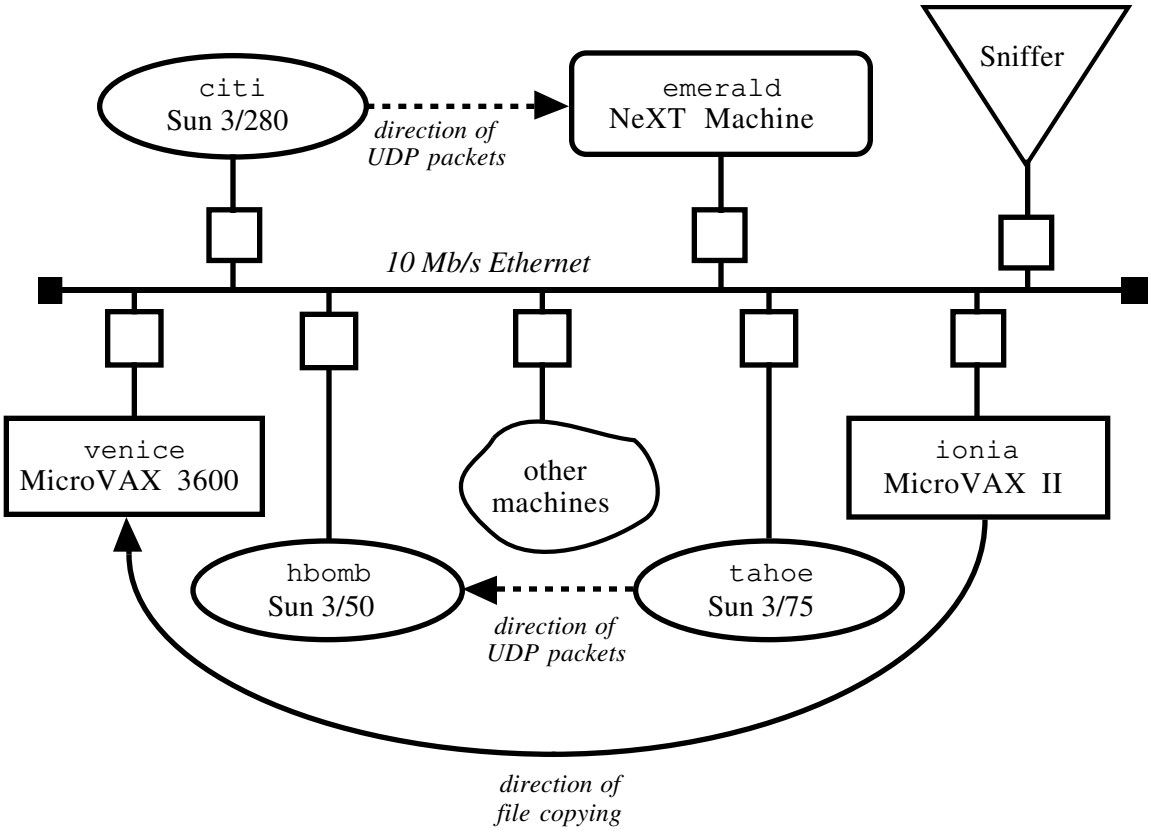


Figure 3 — Background Load Generation Configuration

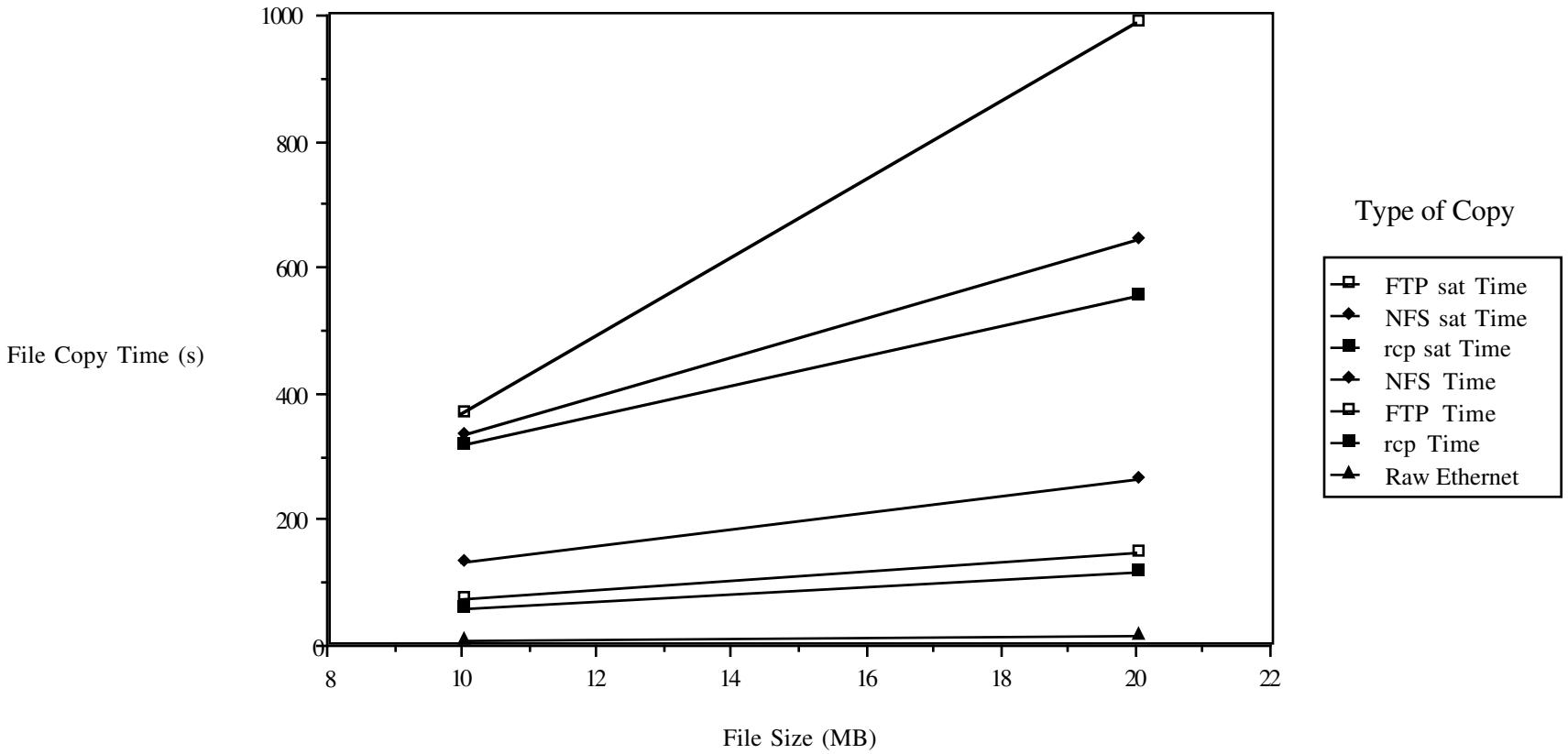


Figure 4 — Unix File Copy Performance on a Saturated Ethernet

In each set of runs for a given experiment, the difference between the lowest and highest copy time was no larger than 20% of the average copy time for that set. Each experiment was performed several times, discarding the lowest and highest times, and using the average of the remaining times for the analysis. In Fig. 4 the FTP copy times were larger than the NFS copy times.

On a procedural note, this set of experiments was very difficult to run because the computers producing and consuming the background traffic constantly crashed from the bursts of UDP packets. The packet bursts forced the machines to run out of kernel memory buffers. By observing and listening[1] to the Sniffer, we could ascertain when a computer crashed and restart the experiment in progress.

4. COMPARISON OF LIVE TEST DATA WITH ANALYTICAL TOOL PREDICTIONS

The Network Modeling Tool (NMT) is an Excel-based tool that is used interactively to create spreadsheets that model the performance of interconnected local computer networks [15]. Earlier file transfer experiments on the token ring model alone was done in [16]. The tool is controlled by the user through customized menus, and when a new model is created, it queries the user via dialogues about the submodels that make up the network and their characteristics. It then builds a spreadsheet with groups of formulas corresponding to each submodel and workstation type, and sections of input parameters that determine the behavior of the network being modeled. For the situation investigated in this paper, a model was built containing one Ethernet, one sending workstation, one receiving workstation, and one workstation representing background traffic. The following is the model built by NMT as it appears on the screen.

---

[1]The Sniffer generates a tone as it monitors network traffic. A higher-frequency tone indicates higher network utilization.

File Transfer Problem				**5/5/89**
Models transfer of large file over ethernet				
Includes background traffic				
Default workstation parameters			**Topology**	
Packet rate (pps per ws)			sending_workstations per Local_	1
Packet size (bits)			receiving_workstations per Loca	1
			background_workstations per Lo	1
Results				
Per-packet delay (ms)			Background traffic	
Total number of packets			Total traffic on Local Ethernet	
Aggregate delay (packets*delay)				
Simple delay (packets/rate)				
Total delay in seconds				
Submodel 1 - Local_Ethernet			**Submodel 2 - sending_workstation**	
Packets from sending_workstations to Local_Ether			Packets leaving sending_workstation	
Packets from receiving_workstations to Local_Eth			sending_workstation mean packet size	
Packets from background_workstations to Local_Ethernet				
Packets to Local_Ethernet			**Submodel 3 - receiving_workstation**	
Packets on Local_Ethernet			Packets leaving receiving_workstation	
Local_Ethernet mean packet size			receiving_workstation mean packet size	
Speed of Local_Ethernet (bps)				
Utilization of Local_Ethernet (data)			**Submodel 4 - background_workstation**	
Utilization (data & collisions)			Packets leaving background_workstation	
Effective throughput of Local_Ethernet (pps)			background_workstation mean packet size	
Delay across Local_Ethernet (ms)				

Figure 5 — NMT Model of the Test Configuration

Note that for each workstation type, we need to know the rate at which it generates traffic, and the average size of the packets it generates. To do this, the monitoring tool `tcpdump` was used to collect statistics for each run of the experiment. Figs. 6-8 show the measured traffic rate and packet sizes for a 50 MB file copy.

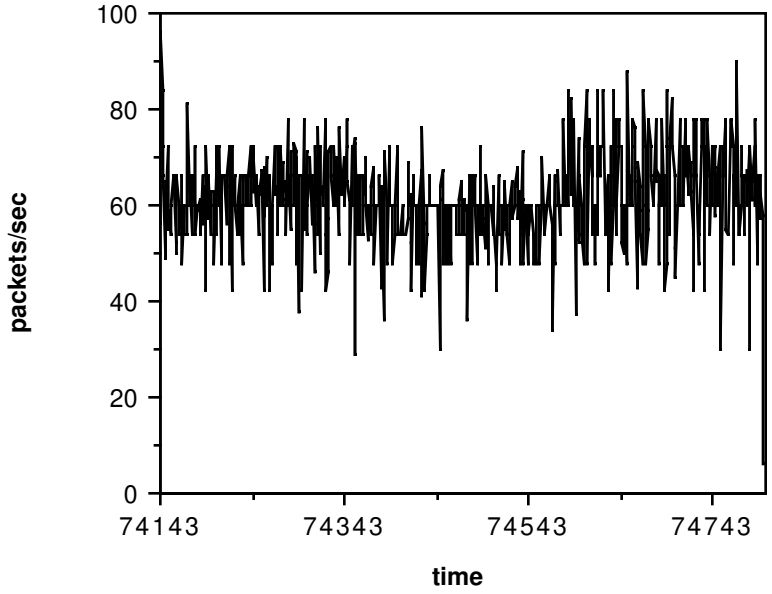


Figure 6 — 50 MB cp: Traffic from `ionia`

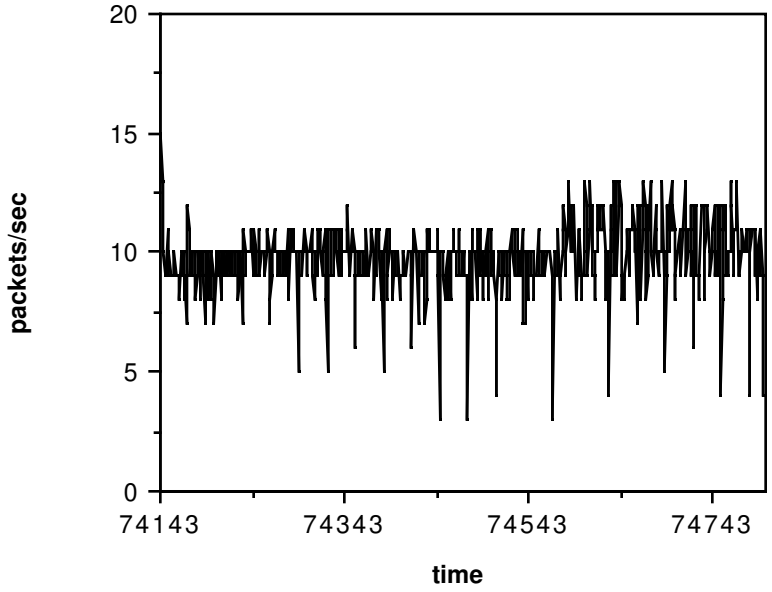


Figure 7 — 50 MB cp: Traffic to `ionia`

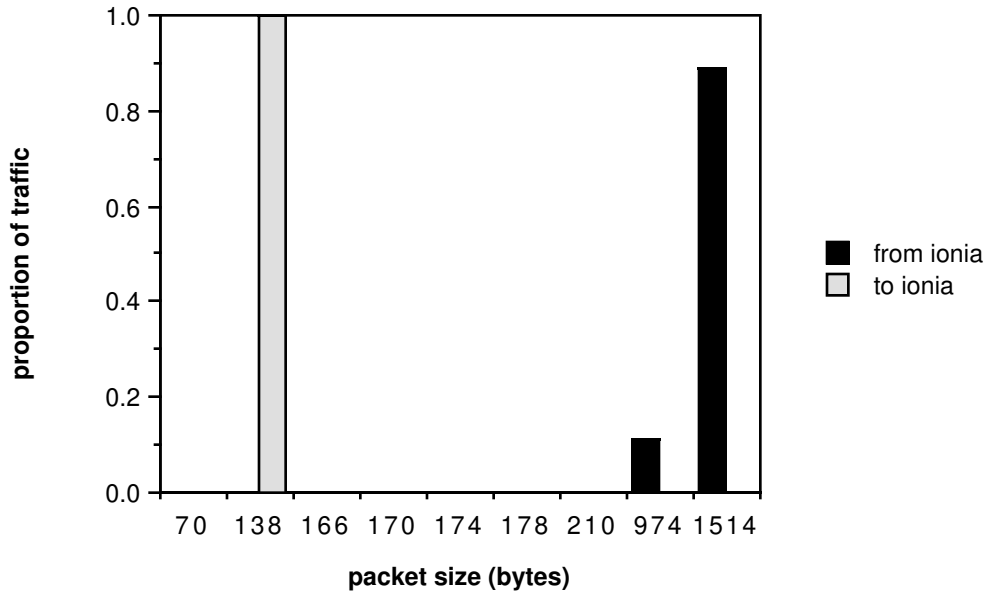


Figure 8 — 50 MB cp from `ionia` to `venice`

As we see, the sending workstation generated an average of 60 packets/sec, with average size 1460 bytes, and the receiving workstation generated an average of 10 packets/sec, all 138 byte acknowledgements. Entering these values into the model, we get the following results for the contention-free case.

File Transfer Problem				**5/5/89**
Models transfer of large file over ethernet				
Includes background traffic				
Default workstation parameters			**Topology**	
Packet rate (pps per ws)	60		sending_workstations per Local_	1
Packet size (bits)	11,680		receiving_workstations per Loca	1
			background_workstations per Lo	1
Results				
Per-packet delay (ms)	1.06671584		Background traffic	0
Total number of packets	68493.1507		Total traffic on Local Ethernet	711,840
Aggregate delay (packets*delay	73.0627286			
Simple delay (packets/rate)	1141.55251			
Total delay in seconds	1141.55251			
Submodel 1 - Local_Ethernet			**Submodel 2 - sending_workstation**	
Packets from sending_workstati	60		Packets leaving sending_worksta	60
Packets from receiving_worksta	10		sending_workstation mean packe	11680
Packets from background_works	0			
Packets to Local_Ethernet	70		**Submodel 3 - receiving_workstation**	
Packets on Local_Ethernet	70		Packets leaving receiving_works	10
Local_Ethernet mean packet size	10169.1429		receiving_workstation mean pac	1104
Speed of Local_Ethernet (bps)	10000000			
Utilization of Local_Ethernet (da	6.64%		**Submodel 4 - background_workstation**	
Utilization (data & collisions)	6.64%		Packets leaving background_wor	0
Effective throughput of Local_Et	65.30		background_workstation mean pa	8376
Delay across Local_Ethernet (ms	1.07			

Figure 9 — Contention-free Ethernet Configuration

It is evident from Fig. 9 that software delays on the sending workstation are much greater than delay attributable to transmission across the network. This can be seen by comparing the software *simple delay* (calculated by dividing the number of packets making up a 100 MB file by the rate at which they are generated) to the network transmission *aggregate delay* (calculated by multiplying the per-packet delay calculated in the Ethernet submodel by the number of packets). Examining the complete set of data for traffic rate each way, it was observed that the rate started out much higher (80-90 packets/sec and 15 packets/sec) and then settled down to the steady-state values observed for the remainder of the experiment (60 and 10 packets/sec respectively). It is reasonable to guess that the higher rate is the rate at which the network software can transfer data until its buffers fill up, and the lower rate is the rate at which the filesystem software can accept data.

Using the model generated by NMT, we can predict at what point the aggregate delay will exceed the simple delay, which is the point at which we would expect to see file transfer time to begin increasing. In the next screen shot (Fig. 10) we see the model with the background traffic increased to 1034 packets/sec, at which point the aggregate delay has just exceeded the simple delay.

File Transfer Problem				**5/5/89**
Models transfer of large file over ethernet				
Includes background traffic				
Default workstation parameters			**Topology**	
Packet rate (pps per ws)	60		sending_workstations per Local_	1
Packet size (bits)	11,680		receiving_workstations per Loca	1
			background_workstations per Lo	1
Results				
Per-packet delay (ms)	17.1082081		Background traffic	8,660,784
Total number of packets	68493.1507		Total traffic on Local Ethernet	9,372,624
Aggregate delay (packets*delay	1171.79507			
Simple delay (packets/rate)	1141.55251			
Total delay in seconds	1171.79507			
Submodel 1 - Local_Ethernet			**Submodel 2 - sending_workstation**	
Packets from sending_workstati	60		Packets leaving sending_workst	60
Packets from receiving_worksts	10		sending_workstation mean packe	11680
Packets from background_works	1034			
Packets to Local_Ethernet	1104		**Submodel 3 - receiving_workstation**	
Packets on Local_Ethernet	1104		Packets leaving receiving_works	10
Local_Ethernet mean packet size	8489.69565		receiving_workstation mean pac	1104
Speed of Local_Ethernet (bps)	10000000			
Utilization of Local_Ethernet (da	48.10%		**Submodel 4 - background_workstation**	
Utilization (data & collisions)	48.10%		Packets leaving background_wor	1034
Effective throughput of Local_Et	566.52		background_workstation mean pa	8376
Delay across Local_Ethernet (ms	17.11			

Figure 10 — Heavy Contention Ethernet Configuration

Note that the average packet size on the Ethernet has decreased, as the average packet size of the background traffic we generated was only 1000 bytes, 2/3 the maximum size. This is the case with the UDP packets that our traffic generator produces. As we can see in the right column of Fig. 10, the model predicts a background traffic level of 87% network capacity, for a total network load of 94% network capacity. The first set of experiments, producing 60% maximum load, falls far short of this, and as would be expected, did not produce a significant difference in delay times. The

second set of experiments, producing near-saturation at 98% load, did increase the delay, as predicted by the model.

Fig. 11 contains a graph comparing the predictions of the model to the measured delay times for the 20 MB file copy. Because the input parameters to the model were based on measurements of NFS file transfers, those are the measured values that the model predictions should be compared to. As we can see, the model is in reasonably close agreement with the measurements for the cases of low, medium and high loading of the network; the measured values range from 16-37% of values predicted from the analytical model. It appears that better accuracy at high loads will require an accounting of the effect of increased transmission delay upon the software delay. One possible source of increased software delay might be the response of RPC's retransmission algorithm to increased transmission delay and occasional packet loss due to multiple collisions.

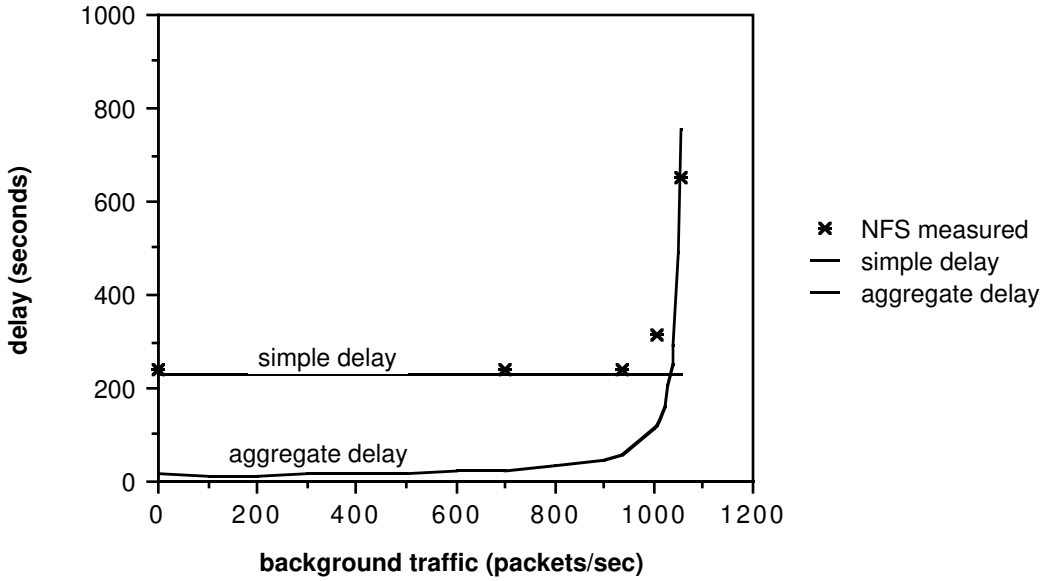


Figure 11 — NMT Predictions and Observed Behavior

5. CONCLUSIONS AND FUTURE WORK

A number of file copy experiments were run under varying network load and measured performance was compared to predictions from an analytic (mean value) model. The experiments show that realistic network loads (normal people doing normal work) do not affect the file copy times. A nearly saturated network will significantly degrade file copy performance, however. The exact performance in either case is partially determined by the type of hardware and the file transfer mechanism used. On our

Ethernet, rcp had the best performance and NFS had the worst. However, since NFS is the easiest to use, the added convenience of NFS may offset its performance penalty for many applications.

We plan to continue to investigate file copy performance under varying network load. A number of recent papers [4],[5] have reported on the causes of delay in the TCP, UDP and IP layers on a Unix workstation. The relative contribution of RPC and disk accesses would complete the picture and allow for a complete accouting of file transfer times. We also intend to explore the effect of varying the *number* of workstations generating background traffic on an Ethernet on file copy performance. These experiments will help us design more accurate models for the Network Modeling Tool, and provide insights into the factors that affect performance of networked distributed computation.

Acknowledgements

The authors grateafully acknowledge the programming and critical review support from Tom Unger and Peter Honeyman.

5. REFERENCES

[1] J. G. Mitchell, and J. Dion "A Comparison of Two Network-Based File Servers," *Communications of the ACM* vol.25, no.4, April 1982, pp. 233-245.

[2] E. D. Lazowska, J. Zahorjan, D. R. Cheriton, and W. Zwenepoel, "File Access Performance of Diskless Workstations," *ACM Trans. on Computer Systems* vol.4, no.3, August 1986, pp. 238-268.

[3] B. Bhargava, T. Mueller, and J. Riedl "Experimental Analysis of Layered Ethernet Software," *Proceedings of the 1987 Fall Joint Computer Conference*, pp. 559-568.

[4] L. Cabrera, E. Hunter, M. J. Karels, and D. A. Mosher "User-Process Communication Performance in Networks of Computers," *IEEE Trans. on Software Eng.* vol.14, no.1, January1988, pp. 38-53.

[5] D. D. Clark, V. Jacobson, J. Romkey, and H. Salwen "An Analysis of TCP Processing Overhead," *IEEE Communications Magazine*, vol. 27, no. 6, June 1989, pp. 23-29.

[6] P. Gunningberg, M. Bjorkman, E. Nordmark, S. Pink, P. Sjodin, and J. Stromquist "Application Protocols and Performance Benchmarks," *IEEE Communications Magazine*, vol. 27, no. 6, June 1989, pp. 30-36.

[7] J. Postel and J. Reynolds "File Transfer Protocol", Network Working Group RFC 959, October 1985.

[8] Sun Microsystems, "NFS: Network File System," Network Working Group RFC 1094, March 1989.

[9] R. Sandberg, D. Goldberg, S. Kleiman, D. Walsh, and B. Lyon "Design and Implementation of the Sun Network Filesystem," *USENIX Summer 1985*, pp. 119-130.

[10] Sun Microsystems "XDR: External Data Representation standard" Network Working Group RFC 1014, June 1987.

[11] Sun Microsystems "RPC: Remote Procedure Call Protocol specification version 2" Network Working Group RFC 1057, June 1988.

[12] J. Postel "User Datagram Protocol," Network Working Group RFC 768, August 1980.

[13] J. Postel "Internet Protocol," Network Working Group RFC 791, September 1981.

[14] J. Postel "Transmission Control Protocol," Network Working Group RFC 793, September 1981.

[15] D. W. Bachmann, M. M. Srinivasan, and T. J. Teorey "Network Modeling Tool: A Large-Scale Campus Network Configurator," CITI Tech. Report 89-2, Univ. of Michigan, June 1989.

[16] P. Martini, O. Spaniol, and T. Welzel "File Transfer in High-Speed Token Ring Networks: Performance Evaluation by Approximate Analysis and Simulation," *IEEE JSAC* vol.6, no.6, July 1988, pp. 987-996.