1994-06

Throughput analysis between high end workstations across an FDDI network

Schivley, Mark A.

Monterey, California. Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

Throughput Analysis Between High End Workstations Across an FDDI Network

by

Mark A. Schivley

June 1994

Thesis Advisor: G.M. Lundy

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY 2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Computer Science Dept. Naval Postgraduate School — 6b. OFFICE SYMBOL (if applicable) CS	7a. NAME OF MONITORING ORGANIZATION Naval Postgraduate School
6c. ADDRESS (City, State, and ZIP Code) Monterey, CA 93943-5000	7b. ADDRESS (City, State, and ZIP Code) Monterey, CA 93943-5000
8a. NAME OF FUNDING/SPONSORING ORGANIZATION — 8b. OFFICE SYMBOL (if applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code)	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)
Throughput Analysis Between High End Workstations Across an FDDI Network (Unclassified)

12. PERSONAL AUTHOR(S)
Schivley, Mark Allen, CPT.

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Master's Thesis	FROM 09/92 TO 06/94	1994, June, 16	159

16. SUPPLEMENTARY NOTATION The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the United States Government.

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	FDDI, TCP/IP

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Recently developed high speed networks are capable of transmitting data at rates of 100 Mbps or more. One such network protocol is Fiber Distributed Data Interface (FDDI). This network has a physical transmission rate of 100 Mbps. Analytical and simulation studies have shown that the FDDI protocol should provide actual throughput of 80% to 95% of this physical rate. Can the end user expect to see this kind of performance? If not, then what kind of throughput can actually be expected and where are the bottle necks?

In order to answer these and other related questions, two areas were studied: First, a performance comparison between a 40MHz SPARCstation 10 workstation and a 50MHz SPARCstation 10 workstation was conducted using the Neal Nelson commercial benchmark tool. Next, a well-known network measurement tool, *ttcp*, was used to obtain data transfer rates while varying several tunable operating system and network parameters. The parameters varied were: Target Token Rotation Time, TCP/IP window size, NFS asynchronous threads, Logical Link buffer size and Maximum Transfer Unit size. The results from the commercial benchmark analysis were used to determine if there are any differences which can affect transfer rates between the two workstations.

The results from the commercial benchmark tool clearly showed that the newer, higher speed processor is faster. The network tool *ttcp* showed that the TCP/IP window size had the largest impact on throughput performance. Throughput more than doubles from a window size of 4k to a window size of 20k. This is followed by having more than one workstation transmitting data simultaneously. Having two workstations transmitting nearly halves throughput. This is followed by having a faster processor. A measurement of file transfers using *rcp* system calls showed that the largest impact on file transfer speed is the overhead of receiving the transferred file.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION
[X] UNCLASSIFIED/UNLIMITED [ ] SAME AS RPT. [ ] DTIC USERS	UNCLASSIFIED

22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE (Include Area Code)	22c. OFFICE SYMBOL
Prof G.M. Lundy	(408) 656-2094/2449	CS/Ln

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

Approved for public release; distribution is unlimited

Throughput Analysis Between High End Workstations Across an FDDI Network

by

Mark A. Schivley
Captain, United States Army

Submitted in partial fulfillment of the requirements for the degree of

MASTER OF COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
June 1994

Author: [redacted]
Mark A. Schivley

Approved By: [redacted]
G.M. Lundy, Thesis Advisor

[redacted]
Shridhar B. Shukla, Second Reader

[redacted]
Ted Lewis, Chairman,
Department of Computer Science

ABSTRACT

Recently developed high speed networks are capable of transmitting data at rates of 100 Mbps or more. One such network protocol is Fiber Distributed Data Interface (FDDI). This network has a physical transmission rate of 100 Mbps. Analytical and simulation studies have shown that the FDDI protocol should provide actual throughput of 80% to 95% of this physical rate. Can the end user expect to see this kind of performance? If not, then what kind of throughput can actually be expected and where are the bottle necks?

In order to answer these and other related questions, two areas were studied: First, a performance comparison between a 40MHz SPARCstation 10 workstation and a 50MHz SPARCstation 10 workstation was conducted using the Neal Nelson commercial benchmark tool. Next, a well-known network measurement tool, *ttcp*, was used to obtain data transfer rates while varying several tunable operating system and network parameters. The parameters varied were: Target Token Rotation Time, TCP/IP window size, NFS asynchronous threads, Logical Link buffer size and Maximum Transfer Unit size. The results from the commercial benchmark analysis were used to determine if there are any differences which can affect transfer rates between the two workstations.

The results from the commercial benchmark tool clearly showed that the newer, higher speed processor is faster. The network tool *ttcp* showed that the TCP/IP window size had the largest impact on throughput performance. Throughput more than doubles from a window size of 4k to a window size of 20k.This is followed by having more than one workstation transmitting data simultaneously. Having two workstations transmitting nearly halves throughput. This is followed by having a faster processor. A measurement of file transfers using *rcp* system calls showed that the largest impact on file transfer speed is the overhead of receiving the transferred file.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

I. INTRODUCTION

A. BACKGROUND

Data communication networks are now an essential part of our society. Our technology base has given us workstations which can process data at speeds which makes mainframes from just a few years ago look slow in comparison. Now, not only must we process the data faster, but we also distribute the information to other locations at speeds which just a few years ago were impossible. We truly are in the information era.

In the 1960s and 1970s, the computer industry worked hard to develop new technologies which would give us faster, more powerful computers. The dramatic advances in integrated circuits technology made possible the wide availability of larger, more powerful super computers, low-cost workstations, and personal computers [ALBE94]. There were the companies which believed that the large, centralized processors were the solution to everyone's problems. At the same time, other companies developed smaller computers called minicomputers. These minicomputers, and their successors, desktop workstations, started filling the needs of small companies and universities which couldn't afford the cost of large mainframes and did not need the processing power provided by the large, all in one solution provided by the mainframe.

In the world of mainframes, the need to distribute data to other computers was not critical. The single mainframe would handle all of a company's processing needs. If there was a need to handle additional processing, the manufacturer of that mainframe provided a solution which would allow their mainframe to communicate with another of their mainframes. This of course ensured that the company or university continued to buy all or most of their computer equipment from the same computer manufacture.

With the growth of the minicomputers and the workstations came the need to connect these less expensive and less powerful machines. This provided the motivation and the driving force behind the development of Local Area Networks (LAN). There were the proprietary options provided by the computer manufactures. However, with the need to provide connectivity between systems came the desire to have connectivity between systems from different manufacturers. This was very difficult without some sort of agreed upon standards. In the late 1970s, the International Standards Organization (ISO) developed the Open Systems Interconnection (OSI) reference model to serve as the basis for future open networks. This model would provide the basis for computers from different vendors to be able to communicate with each other [ALBE94].

Now we have the beginnings of connectivity between computers and the beginnings of smaller, more powerful computers. In the 1980s, Sun Microsystems started producing their line of desktop workstations. Within a few years, these workstations were being based on new Reduced Instruction Set Computer (RISC) technology which allowed Sun Microsystems and other companies to produce faster, more powerful workstations. Now if we combine the advancements of the desktop workstations with the advancements made in networks, we have the true beginnings of the information era.

The question now becomes one of which technology is advancing faster. Are we producing workstations which can exceed the capability of the networks or are the networks staying ahead of the abilities of the workstations. Also, advancements in workstation technology isn't just limited to faster hardware. Is the operating system and its networking tools keeping pace with current demands?

It is clear that the workstations are faster and more powerful than in the past. It is also clear that the networks can handle more data at faster rates than in the past. But where do we stand if we compare a recently released product produced by Sun Microsystems with one of the current high speed networks such as Fiber Distributed Data Interface (FDDI)?

B. OBJECTIVE

The objective of this thesis will be to measure actual throughput between high performance workstations over an FDDI network to determine what bottlenecks, if any, exits between Sun Microsystem SPARCstation™ 10 multiprocessors running Solaris™ 2.3 and the Network Peripheral™ SBus FDDI Network Interface cards and to evaluate Transmission Control Protocol/Internet Protocol (TCP/IP) as a high speed transport protocol. This process will require an analysis of the workstations being used in this study, an understanding of current network operating system tools and measurements of data transfers across the network being tested.

This is not simply a matter of reading the vendor's promotional literature and seeing which aspect of the distributed processing environment is more capable. Vendors normally promote those aspects of their products which they can demonstrate as performing at or above some threshold. This threshold may or may not be value to the consumer.

C. SCOPE, LIMITATIONS AND ASSUMPTIONS

The scope of this investigation is limited to performing testing and tuning at the level available to any system administrator. No modifications are made to any hardware or changes made to the workstation kernel which are not considered tunable parameters. From this investigation, a determination will be made as to whether or not there are any bottlenecks.

It is assumed that the changes made and the results observed on the SPARC 10 multiprocessors running Solaris 2.3 can be extrapolated to other vendor's hardware and software. If we note that changing the TCP/IP window size on our workstations results in a 10 fold increase in throughput, then we assume comparable results would be observed on other vendor's workstations.

D. ORGANIZATION OF THESIS

This thesis is organized into seven chapters. This chapter provides the introduction and scope of work to be performed. Chapters II and III provide a background on networks in general, FDDI specifically and the specifics on the workstations involved in this investigation. Chapters IV and V cover the methodology, test results and analysis of results. Chapter VI covers what conclusions can be derived from these results.

II. NETWORK PROTOCOLS

A. NETWORKING THEORY

The primary focus behind the development of network protocols has been the organization of the protocol into a series of layers. This has allowed the design of the protocols to be simplified by focusing attention at each layer upon that layer's function and its interaction with the layers above and below. The purpose of each layer is to offer certain services to the layer above without the higher layer needing to know how those services were provided.

When designing a network protocol the network designer must determine how many layers the protocol will have, what those layers will do and how the layers will communicate with each other. This last decision, deciding how the layers will communicate, is one of the more important considerations. A clean-cut interface must be defined which will minimize the amount of information that must be passed between layers.

The set of layers and protocols is know as the network architecture. Enough specification must be given for each layer of the protocols so that vendors can write their versions of the protocol for their computer architecture. This is what makes the network architectures beneficial to everyone accessing a network. By having an agreed upon network architecture that everyone is willing to use, we can have distributed processing over heterogeneous processors [MINO91].

B. OPEN SYSTEM INTERCONNCETION

The Open System Interconnection (OSI) reference model, Figure 1, was proposed in 1978 to promote compatibility between network designs. This model was approved as a standard [ALBE94] in 1983 by the International Standards Organization (ISO). The reference model is not a protocol or set of rules but a layering of required functions, or services, that provides a framework with which to define protocols. In practical terms, OSI is seen as a means of developing communications networks which are not restricted by the need to conform to a rigid set of manufactures' proprietary standards and protocols.

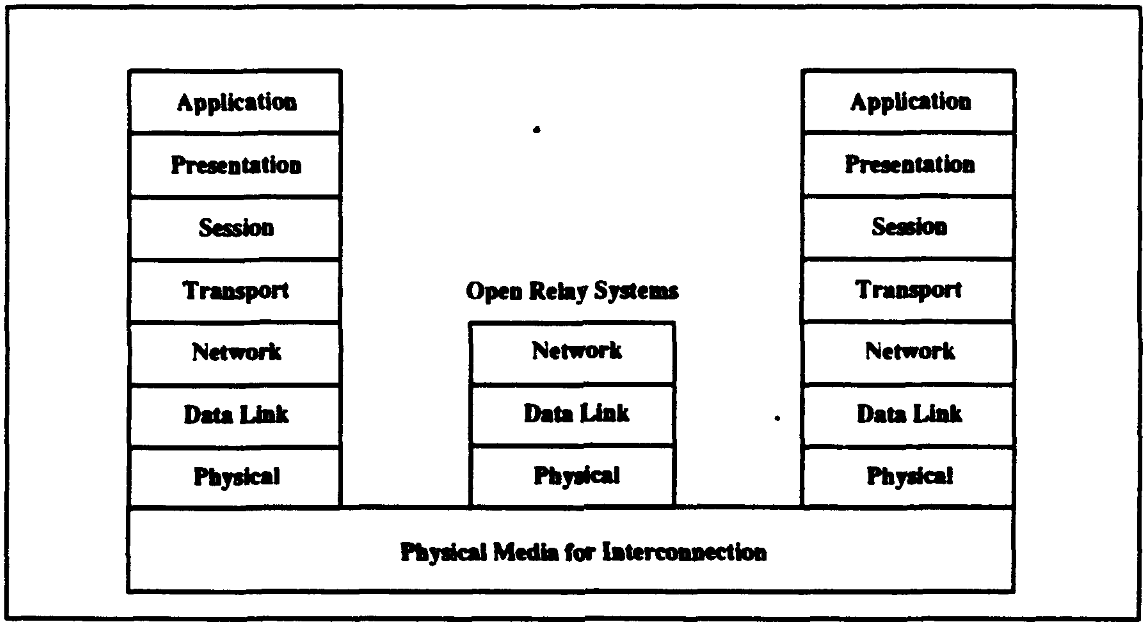

Figure 1: ISO-OSI Reference Model

The purpose of these seven layers is to define the various functions that must be carried out when two machines communicate. Each of the seven layers is architecturally independent, so that the relevant protocols and service functions of each layer can be developed independently. The seven layers of the model can be roughly divided into two parts; the first four layers, physical to transport, provide the telecommunications functions and operate on a node-to-node basis. The top three layers, session to application, are concerned mainly with carrying out processing functions and creating a meaningful dialog between the user and the application.

Below are the seven layers of the OSI model [STAL91]:

- Layer 1: Physical Layer
- Layer 2: Data Link Layer
- Layer 3: Network Layer

- Layer 4: Transport Layer
- Layer 5: Session Layer
- Layer 6: Presentation Layer
- Layer 7: Application Layer

C. TRANSMISSION CONTROL PROTOCOL/INTERNET PROTOCOL

The Transmission Control Protocol/Internet Protocol (TCP/IP) protocol is also structured as a series of layers. Each layer is designed for a specific purpose. They are designed so that a specific layer on one machine sends or receives exactly the same object sent or received by its twin on another machine. This is done without regard to what is going on in layers above or below the layer under consideration.

The advantage of layering is that it simplifies protocol design. The designer can concentrate on a specific layer without regard to the design of other layers. For example, when designing the transport layer of the protocol, the engineer need be concerned only with assuring that a packet received by one machine is identical to the packet sent by another. The message contained in the packet is of no concern. The integrity of the message is of concern only to the designer of the application layer.

Members of the TCP/IP family include the Internet Protocol (IP), Transmission Control Protocol (TCP), User Datagram Protocol (UDP), Address Resolution Protocol (ARP), Reverse Address Resolution Protocol (RARP), and the Internet Control Message Protocol (ICMP). The entire family may be referred to as TCP/IP, reflecting the names of the two main protocols.

The OSI model describes an idealized network communications model. TCP/IP does not correspond to this model at every level, but instead either combines the functions of several OSI layers into a single layer, or finds no need to make use of certain layers. In consequence, TCP/IP can be described by a simpler model as shown in Figure 2 [STEV94].

1. Link Layer

The Link layer is the hardware level of the protocol model. It specifies the physical connections between hosts and networks, and the procedures used to transfer packets between machines.

Application	Telnet, FTP, e-mail, etc.
Transport	TCP, UDP
Network	IP, ICMP, IGMP
Link	device driver and interface card

Figure 2: The Four Layers of the TCP/IP Protocol Suite

2. Network Layer

This layer is responsible for machine-to-machine communications. It determines the path a transmission must take, based on the receiving machine's IP address. The network layer also provides transmission formatting services; it assembles data for transmission into an internet datagram. If the datagram is outgoing (received from the higher layer protocols), the network layer attaches an IP header (Figure 3) to it. This header contains a number of parameters, most significantly the IP addresses of the sending and receiving host. Other parameters include datagram length and identifying information, in case the datagram exceeds the allowable byte size for network packets and must be fragmented.

3. Transport Layer

The transport layer protocols enable communications between application programs running on separate machines. The transport layer assures that data arrives in sequence, and without error. It does so by swapping acknowledgments of data reception, and the retransmission of lost packets. This type of communication is known as "end-to-end". Protocols at this level are TCP, UDP, and ICMP.

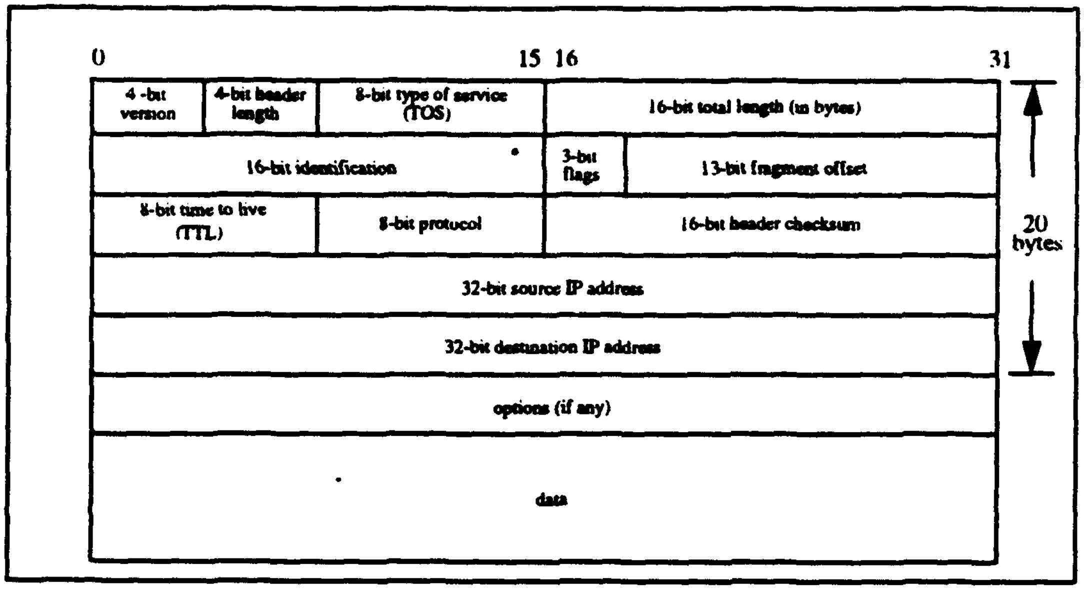

Figure 3: IP Header

TCP attaches a header onto the transmitted data. This header contains a large number of parameters, see Figure 4, which help processes on the sending machine connect to peer processes on the receiving machine. TCP uses 16 bit port numbers as its addressing method. Servers are normally know by their well-known port number. For example, every TCP/IP implementation that provides an FTP server provides that service on TCP port 21. Every Telnet server is on TCP port 23 [STEV94].

4. Application Layer

The application layer lets you use various TCP/IP standard internet services. These services work with the next lowest level of protocols (transport) to send and receive data. These services include *telnet*, *ftp*, *rcp*, and the Domain Name Service (DNS).

telnet. The Telnet protocol enables terminals and terminal oriented processes to communicate on a network running TCP/IP.

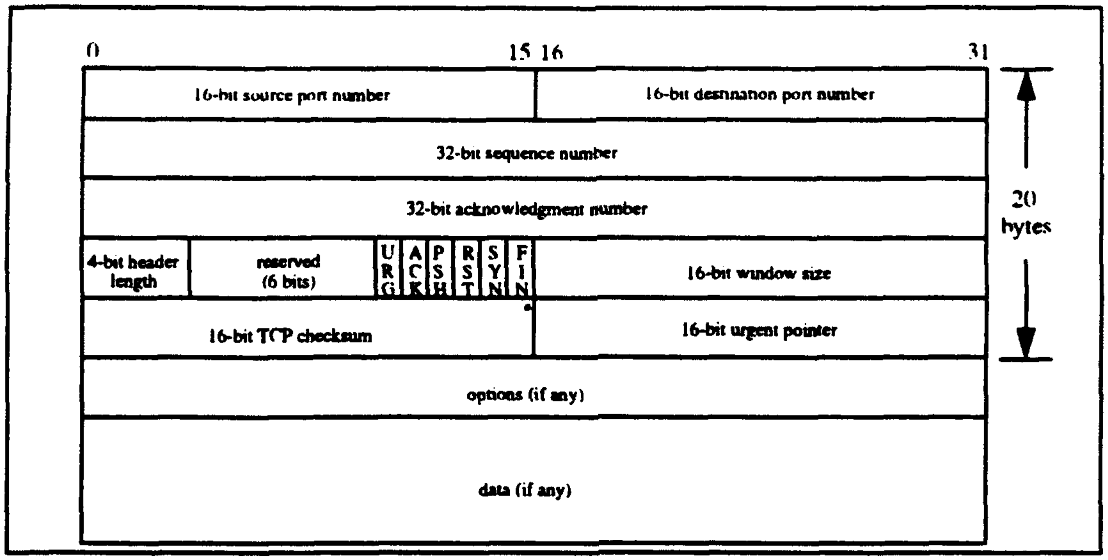

Figure 4: TCP Header

ftp. *ftp* transfers files to and from a remote network. Unlike *rcp*, *ftp* works even when the remote computer is running a non-UNIX operating system. A user must "log in" to the remote computer to make an *ftp* connection unless a system administrator has set up the computer to allow "anonymous ftp".

rcp. *rcp* copies one or more files or hierarchies to and from a remote computer. The remote computer must be running UNIX. One must be an accepted user of the remote computer (i.e., the user's name must be in the remote computer's password database, and the user's machine name must be listed in the remote *.rhost* file). If this is not the case, a user cannot copy anything to or from the remote machine. The user must know the complete pathname of the file or directory to be copied.

DNS. DNS provides host names to the IP address service. It is a distributed database that is used by TCP/IP applications to map between hostnames and IP addresses. The DNS provides the protocol that allows clients and servers to communicate with each other and to provide electronic mail routing information.

D. FIBER DISTRIBUTED DATA INTERFACE

1. Fiber Distributed Data Interface Basics

Fiber Distributed Data Interface (FDDI) is a 100 Mbps high speed LAN standard developed under the auspices of American National Standards Institute (ANSI) X3T9.5 committee. FDDI was developed to create a reliable fault-tolerant, high-speed network connecting numerous stations over greater distances than existing standards. Although FDDI is somewhat similar to the IEEE 802 standards, it is not part of that family of standards [MINO91].

The ANSI X3T9.5 committee developed specifications for a network based on a dual counter-rotating fiber optic ring using a timed-token protocol, which is capable of transmitting data at 100 Mbps in each ring and which can extend to 500 stations over total fiber length of 200 km with full system performance. The dual counter-rotating ring can support connections up to 2 km with multimode fiber and connections up to 60 km using single-mode fiber.

The FDDI standard allows for two types of traffic: synchronous and asynchronous. Synchronous traffic should consist of data which is time sensitive such as voice or interactive video. Any delay in the throughput of this traffic has an adverse affect of the quality of the data being transferred. Asynchronous traffic should consist of more routine data transfers such as email, file transfers and Network File System (NFS) or Network Information Service (NIS) traffic. These packets of data can sustain some reasonable delays in transmission without any adverse affects on the applications.

2. Fiber Distributed Data Interface Layers

The standard for FDDI developed by the X3T9.5 committee included four layers shown in Figure 5. They are the Media Access control (MAC) layer, the Physical (PHY) layer, the Physical Medium Dependent (PMD) layer, and the Station Management (SMT) document [ALBE94].

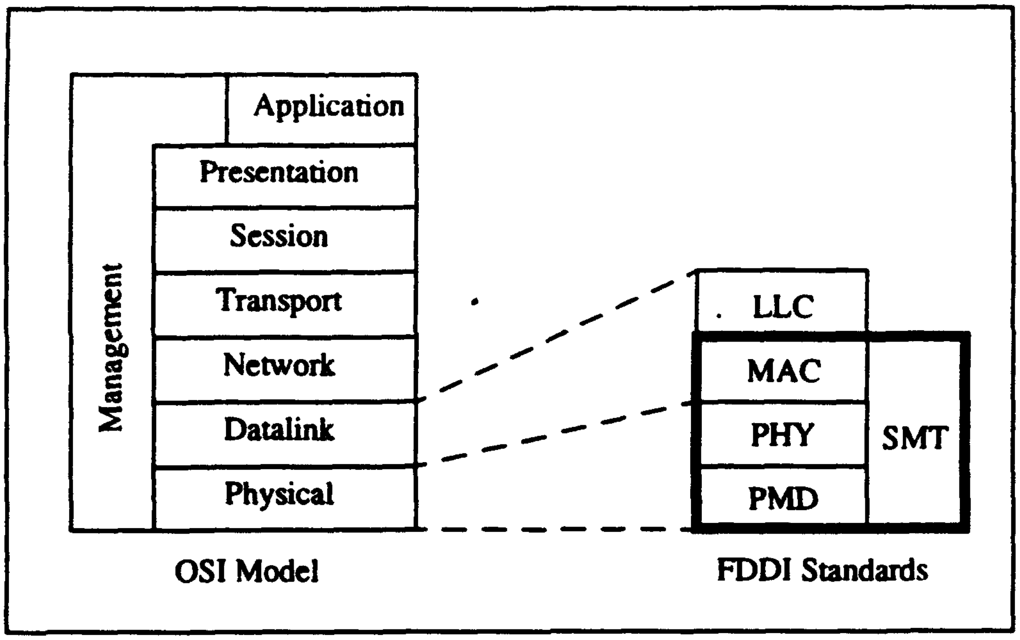

Figure 5: Relationship Between FDDI and ISO-OSI Layers

The four layers of FDDI fall under the first two layers of the OSI Model. The physical layer of FDDI is specified in two documents: the FDDI PMD which defines the optical interconnecting components used to form links and the FDDI PHY which defines the encoding scheme used to represent data and control symbols. The DLL is also divided into two sublayers: A MAC and LLC layer. The MAC portion provides access to the medium, address recognition, and generation and verification of frame check sequences. The LLC specification is not part of the FDDI standard [MINO91].

Below in Figure 6 is an additional graphical representation of the interaction between the FDDI standards as described in [POWE93].

a. The Physical Medium Dependent Layer

This layer defines all transmitters, receivers, cables, connectors and other physical media and hardware. There are currently 6 media options provided for the PMD layer:

• Multimode fiber (PMD)
• Single-mode fiber (SMF-PMD)
• Low-cost fiber (LCF-PMD)
• Shielded twisted pair (STP-PMD)
• Unshielded twisted pair (UTP-PMD)
• FDDI on Synchronous Optical Network (SONET)

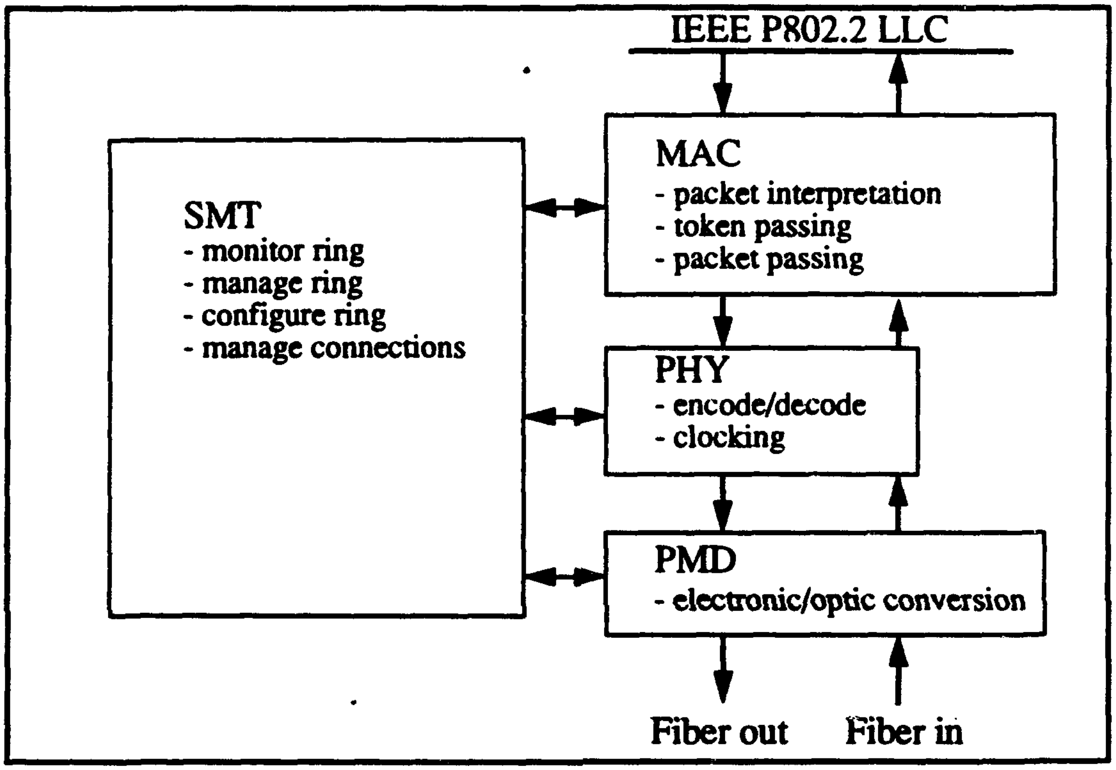

Figure 6: Block Diagram of the FDDI Layers

The first three options are published or soon to be published standards. The last three options are under development [ALBE94].

The PMD layer provides the PHY layer all the services required to transport a coded bit stream from one node to the next node. It converts the encoded data requests from the PHY layer into either optical or electrical signals depending on the media being used. It also provides SMT with the needed services required for proper ring management. The PMD layer informs both the SMT and PHY layers whenever it detects a signal on the medium [ALBE94].

b. The Physical Layer

This layer provides media independent functions associated with the OSI physical layer. The PHY layer decodes incoming bit stream into a symbol stream for use by the MAC layer and it encodes the data and control symbols provided by the MAC layer for transmission via the PMD layer. The PHY layer continuously monitors the ring status by listening to incoming signals and passes this information onto the SMT layer [ALBE94].

c. The Media Access Control Layer

This layer provides fair and deterministic access to the network. The access is fair because a workstation's physical location does not give it any advantage in accessing the medium over another workstation's location. The service is deterministic implies that the time the workstation has to wait for the token can be predicted under error free conditions.

In FDDI, medium access is controlled by a token. The workstation which possesses the token can transmit frames. The other workstations on the network repeat the frame, and the destination workstation copies the frame in addition to repeating it. The MAC layer of the workstation which generated the frame is responsible for removing the frame and passing the token downstream to the next workstation when it's Token Holding Time (THT) has expired [ALBE94].

d. The Station Management Layer

The SMT layer provides services such as node initialization, bypassing faulty nodes, coordination of node insertion and removal, fault isolation and recovery and collection of statistics.The SMT layer provides these functions using services provided by the PMD, PHY and MAC layers.

3. Fiber Distributed Data Interface Framing

Most communications within FDDI is done on frames (Except Physical Connection Management (PCM) signaling). Within the MAC layer there are three frame types:

- Tokens
- Management frames
- Data frames

Each frame is made up of three parts. The first part is the start of the frame sequence. The next part is the data or information part of the frame. The last part is the end of the frame sequence. The data frame is shown in Figure 7 along with the size of each field in symbols [ALBE94].

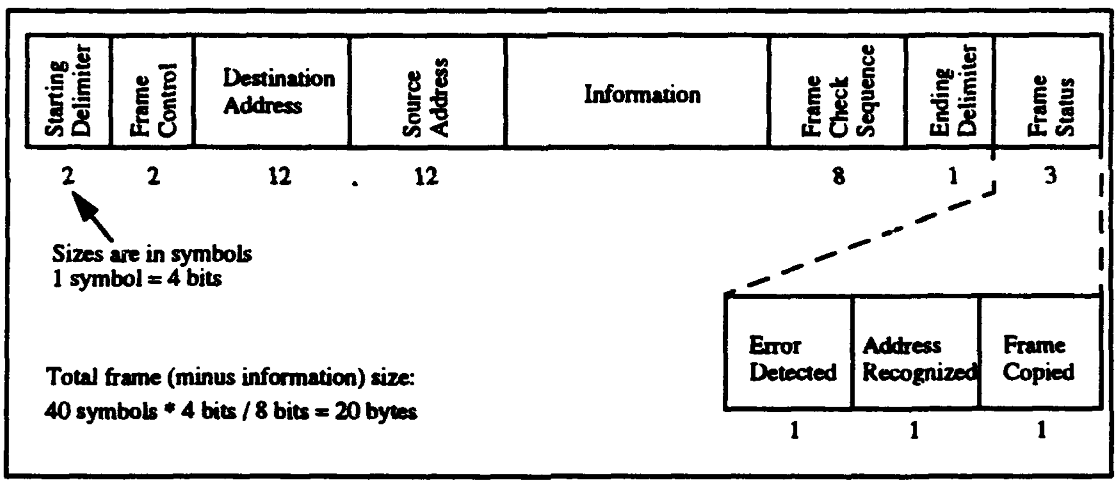

Figure 7: FDDI Frame Format

The start part of the frame is 28 symbols in length. Each symbol is a 4 bit unit. This means the start portion of the FDDI frame is 28 symbols * 4 bits / 8 bits = 14 bytes long. The end portion of the FDDI frame is 12 symbols or 6 bytes long. Since the maximum frame length is 9,000 symbols or 4,500 bytes, this leaves 4,480 bytes available for data or information. This remaining portion of 4,480 bytes, is also know as the FDDI Maximum Transfer Unit (MTU) value [ALBE94].

4. Encoding Method

Digital data needs to be encoded for proper transmission.The type of encoding used is determined by the type of media being used, the desired data rate, noise present on the transmission media and other factors. Since FDDI was originally intended for use over fiber optics, the encoding method selected needed to provide a digital-to-analog capability.

FDDI uses a two-stage encoding scheme; 4B/5B group encoding along with the digital signal encoding method known as Non-Return to Zero Inverted (NRZI). NRZI is an example of differential encoding. The signal is decoded by comparing the polarity of adjacent signal elements rather than determining the absolute value of a signal element. In 4B/5B, the encoding is done 4 bits at a time resulting 5 encoded bits. Then, each element of the 4B/5B stream is treated as a binary value and encoded using NRZI.

The result is that FDDI is able to achieve a 100 Mbps throughput using a 125-MHz rate. As mentioned earlier, the PHY layer is responsible for decoding the 4B/5B NRZI signal from the network into symbols that can be recognized by the station. The synchronization is derived from the incoming signal and the data are then retimed to an internal clock through an elasticity buffer.

E. NETWORK OVERHEAD

The process of transferring data from one workstation to another involves all the layers of protocols described previously. Even though the protocols are broken into layers to distribute functionality, the result is increased overhead. As discussed earlier, for each layer of protocol, there is an associated overhead at that layer as shown in Figure 8.

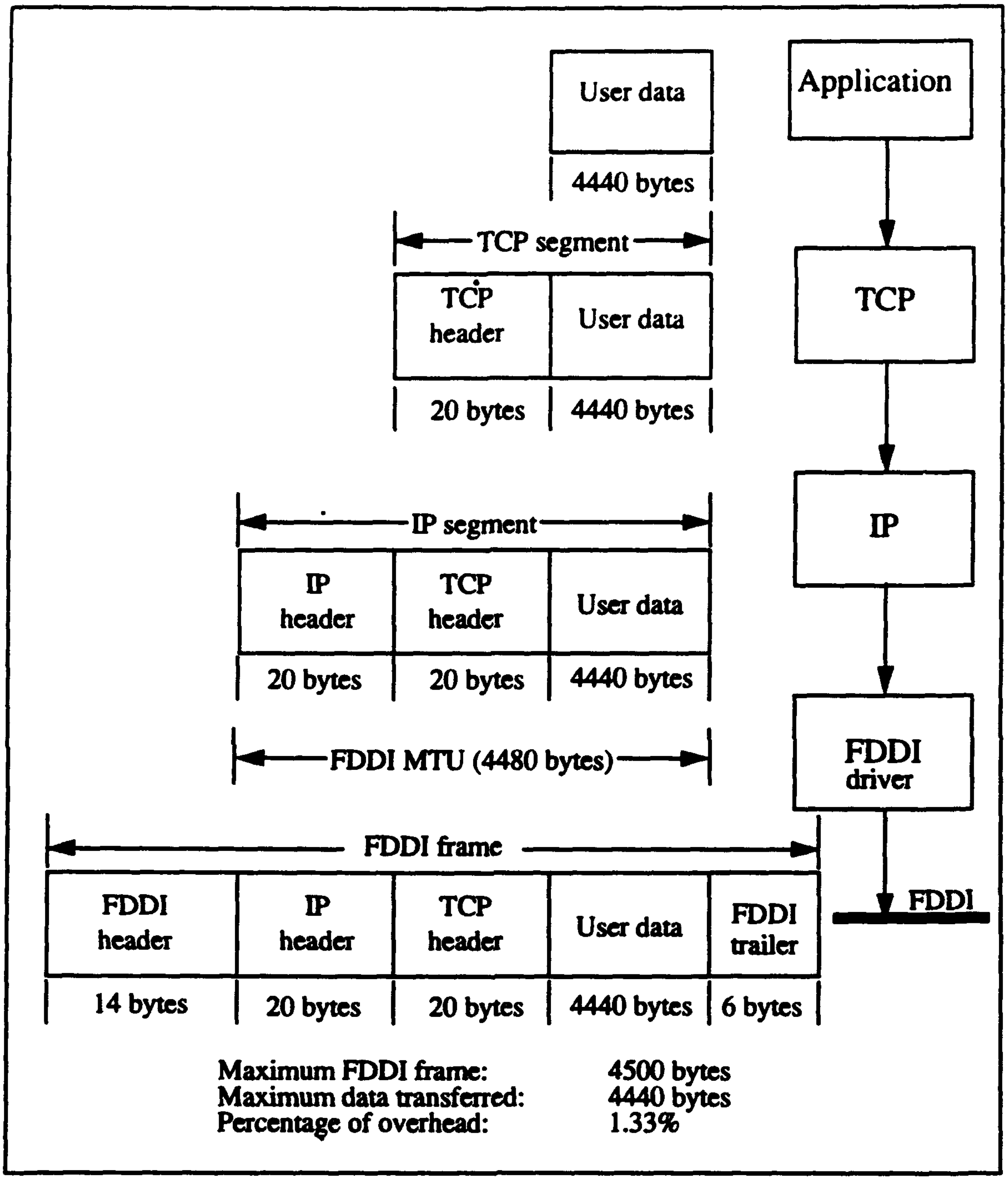

Figure 8: Composition of FDDI Frames and Percentage of Overhead

The amount of overhead involved in transferring data is dependent upon the protocols used and the network media being used as the transfer agent. For FDDI, the overhead is calculated as follows:

Data	Overhead	Level	Total Overhead
4,440 bytes	0	Application	0 bytes
4,440 bytes	20 bytes	TCP	20 bytes
4,440 bytes	20 bytes	IP	40 bytes
4,440 bytes	20 bytes	FDDI	60 bytes

In this example, the frame of data being sent is 4,500 bytes: total amount of data being transferred is 4,440 bytes and total amount of overhead is 60 bytes. Therefore, the percentage of overhead is the amount of overhead (60 bytes) divided by the total frame size (4,500 bytes). Overhead = 60 bytes / 4,500 bytes = 1.33%. If we were to only send 11 bytes of data, then the overhead would be 60 bytes / 71 bytes = 84.5%. It is clear that the more data sent in each FDDI frame, the lower the percentage of overhead associated with that frame. Note that in this example the overhead from the application layer was not included.

F. FIBER DATA DISTRIBUTED INTERFACE PARAMETERS

This section will give a brief explanation of FDDI parameters as covered in the ANSI standards. The MAC layer must implement a number of these parameters as timers and counters. The three main goals of these timers and counters are to [ALBE94]:

- Allow the initialization of the token rotation timer
- Permit fast recovery from ring errors
- Aid in the collection of ring statistics for SMT

Below in Figure 9 are a list of the important timer values and variables used in the data transmission process. According to the FDDI standards, every time a node releases a token, it loads the value of *T_Opr* into Token Rotation Timer (TRT). This timer then decrements until it reaches zero. If it reaches zero before a valid token is received, the token is said to be late and the late counter (*Late_Ct*) is incremented. If TRT expires a second time before a valid token is received, an error condition exists and recovery procedures are initiated.

The token holding timer (THT) is used to control asynchronous transmission in a dynamic manner. When a valid token is received and the *Late_Ct* is not set, the token is said to be early and the node may transmit asynchronous data. In this case, THT is set to *T_Opr* minus TRT and the node may transmit until THT expiries. TVX is a hardware backup timer that is used to prevent nodes from blabbering on the network due to some error or miscalculation of THT [ALBE94].

Parameter	Description
TTRT	Target token rotation time
TRT	Token rotation timer
T_Opr	Operative TTRT negotiated during claim process
Late_Ct	Late counter
THT	Token holding timer
TVX	Transmission valid timer

Figure 9: Timers and Counters Used in Data Transmission

III. NETWORK EQUIPMENT

A. NETWORK OVERVIEW

The Naval Postgraduate School (NPS) FDDI research network consist of the three machines operating on a ring. The names of the three machines on the FDDI LAN are "Black", "White" and "Gold". Gold is the server on the network. The network is setup as shown in Figure 10.

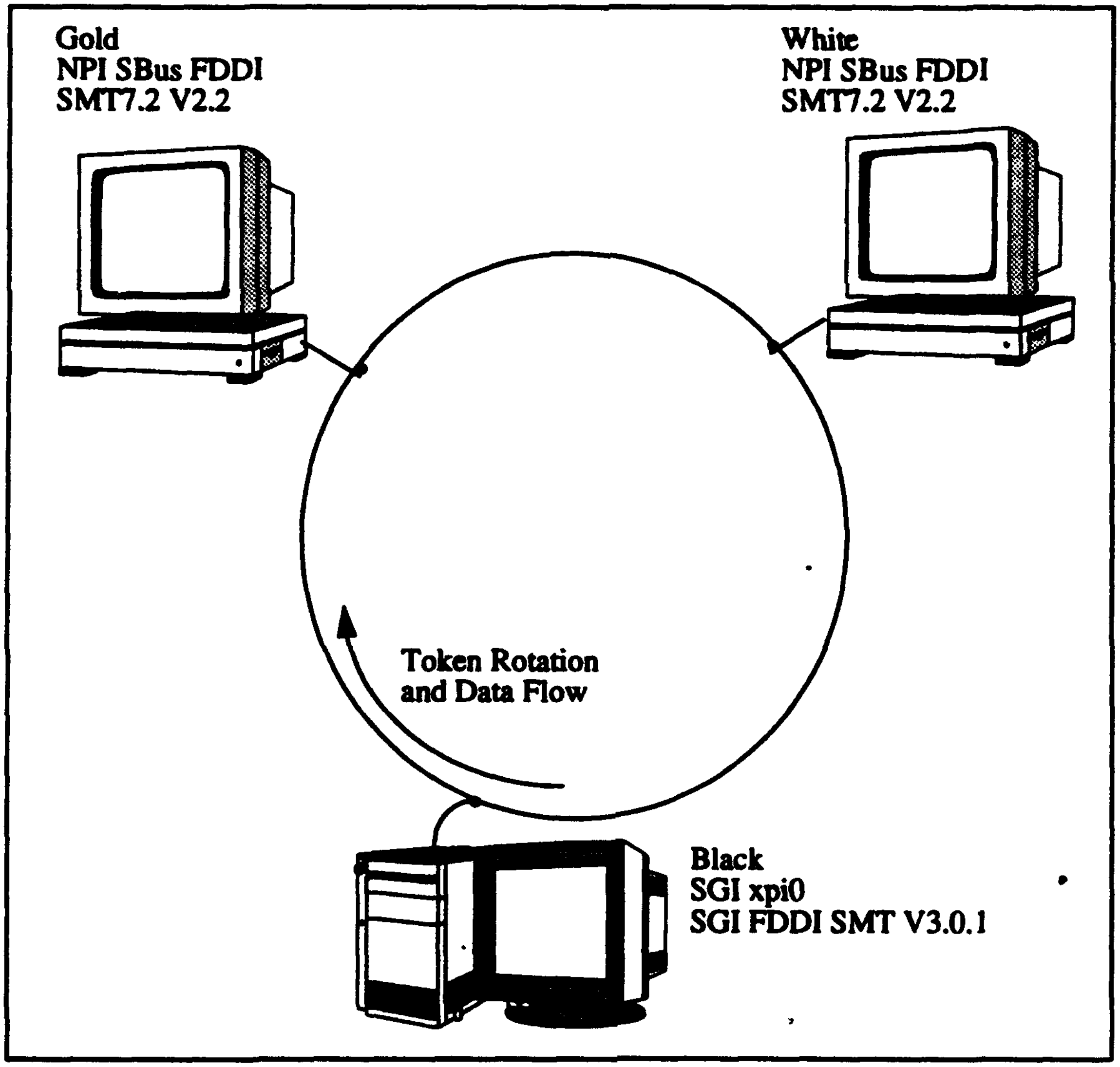

Figure 10: NPS's FDDI Research Network

1. Fiber Optics Equipment

The specifications for the fiber optics equipment can be found in the PMD standards. Originally, only optical fiber was specified as a physical media for FDDI. Now it is possible to also use shielded twisted-wire for short-distance transmissions. The requirements for twisted-wire can be found in the STP-PMD standards.

The recommended fiber size for FDDI is 62.5/125 μ m.The operating wavelength is specified as 1300 nm and the minimum allowable power for the transmitter is -16 dBm. Pin diodes are to be used in the link. Pin diodes were chosen over avalanche photodiodes since pin diodes are a more mature technology and would result in a lower cost receiver. The bit-error rate (BER) of the network is 4 x 10^{-11} and the maximum number of nodes is 500 [POWE93].

2. Network Peripherals' Interface.

The Network Peripherals Inc. (NPI) SBus FDDI Network Interface conforms to Sun Microsystems' requirements for an SBus adapter. It mounts in a SBus slot and implements burst mode Direct Memory Access (DMA) for the highest system performance [NPI93].

As stated earlier, FDDI is designed to provide the capability for both synchronous and asynchronous data transfer. This is not the case with NPI's SBus FDDI Interface card. Furthermore, it is not the case for all known current implementations of FDDI. This makes the relationship of the timers and counters described earlier not as well defined. Without synchronous and asynchronous transfers, there is no need for *Late_Ct* and THT. Below is a list of parameters which NPI list as its tunable parameters. Note that there is not a parameter listed here which specifies how long a node can maintain the token.

sbf_num_llc_rx	/* For LLC network traffic: /* number of 4k receive buffers, maximum is 64 4k buffers /* Default is 48 4k buffers per NP-SB adapter
sbf_num_smt_rx	/* For SMT network traffic:

/* number of 4k receive buffers, maximum is 64 4k buffers
/* Default is 4 4k buffers per NP-SB adapter

sbf_mtu /* Maximum protocol packet size, default is 4352 bytes

sbf_T_Notify /* SMT Neighbor Notification Timer, default is 30 seconds

sbf_num_mcast /* number of multicast entries, default is 16

These parameters can be tuned by entering the appropriate line below in /etc/system for each parameter.

1. To change number of receive buffers to 64:

   set *sbf:sbf_num_llc_rx* = 64

2. To change MTU size to 4192 bytes:

   set *sbf:sbf_mtu* = 4192

3. To change T_Notify timer to 10 seconds:

   set *sbf:sbf_T_Notify* = 10

After contacting NPI it was learned that there is another parameter which is not advertised called *t_req*. This parameter determines how long the node is allowed to hold the token.

3. Silicon Graphic's Interface

FDDIXPress™ 3.0.1 is a network interface controller (board and software) providing FDDI connectivity for Silicon Graphics workstations and servers. For the IRIS Indigo, FDDIXPress has two configurations of the FDDI board: FDDIXPI and FDDIXPID. The FDDIXPI board allows one single-attachment FDDI connection to an FDDI concentrator; the FDDIXPID board provides a dual-attachment FDDI connection directly to the dual ring, or one or two connections to an FDDI concentrator. An Indigo can accommodate one of these boards.

When FDDIXPress is installed, an Indigo can also use its built-in Ethernet network interface, thus having two network interfaces. FDDIXPress for IRIS Indigo has been designed for customer installation.

B. WORKSTATION OVERVIEW

1. SUN SPARCstation 10 system

The SPARCstation 10 systems used in this test were the new multiprocessing systems running Solaris 2.3. We had two SPARCstation 10 systems, Gold and White, available for our FDDI research. Both systems have two processors, two internal hard disk drives and 224 Dynamic Random Access Memory (DRAM). Gold has two 50MHz processors and 2 - 1 GB internal drives. White has two 40MHz processors, 1 -1 GB internal drive and 1-425 MB internal drive.

a. Software Architecture

Solaris 2.3 is a multilayered operating system that includes SunOS 5.3, Open Network Computing (ONC), Open Windows, and the DeskSet. At the core of Solaris is SunOS, the collection of programs that actually manages the system, which includes the kernel, the file system, and the shells.

SunOS is a collection of UNIX programs that control the Sun workstation and provide a link between the user, the workstation, and its resources. It has its roots firmly placed in the two most popular UNIX families: Berkeley UNIX (BSD) and AT&T's UNIX. Early versions of SunOS blended some of AT&T's UNIX with Berkeley UNIX and offered additional enhancements.

AT&T and Sun Microsystems later worked together to create a new industry standard, AT&T UNIX System V Release 4, commonly known as SVR4. SunOS 5.3 merges SunOS 4.1 and SVR4. Most of the new changes in SunOS come from SVR4. As a result, Solaris 2.3 is based on SVR4 but contains a few additional BSD/SunOS features [HESL93].

b. Hardware Architecture

The SPARCstation 10 architecture is shown in Figure 11 [SUNM90]:

SuperSPARC microprocessor This is a high-performance CPU chip that has the following features:

- A single chip with integer, floating point, memory management, and caches.
- Superscalar pipeline with up to three instructions launched per clock cycle.
- 20-Kbyte instruction cache and 16-Kbyte data cache.
- 64 entry TLB with hardware page-table walking.
- Integral support for cache-coherent multiprocessing.

The SuperSPARC processor has a companion chip, the SuperCache controller, which provides for a 1-Mbyte external cache. Additionally, SPARC modules with SuperCache controllers can operate asynchronous to the system clock.

MBus. The MBus is a high performance memory bus which was first introduced in Sun's SPARCserver 600MP family. It is a synchronous, 40-MHz 64-bit bus that is capable of a peak transfer rate of 320 Mbytes/second. Typically, the MBus can sustain a rate of 100 Mbytes/second.

This bus provides support for symmetric multiprocessing by means of a "snooping" protocol. Whenever a processor puts an address onto the MBus, all other processors "snoop" the bus, checking to see if data at the snooped address is in their cache.

Main memory architecture: The Sun-4m architecture uses a 144 bit wide memory data path (128 bits of data and 16 bits of error detection and correction). The use of a 128-bit wide memory data has two advantages. First, the 32-byte cache fill can be accomplished quickly. Second, error corrections can be performed on each 64-bit word. Single bit errors can be corrected and double-bit (4-bit) errors can be detected.

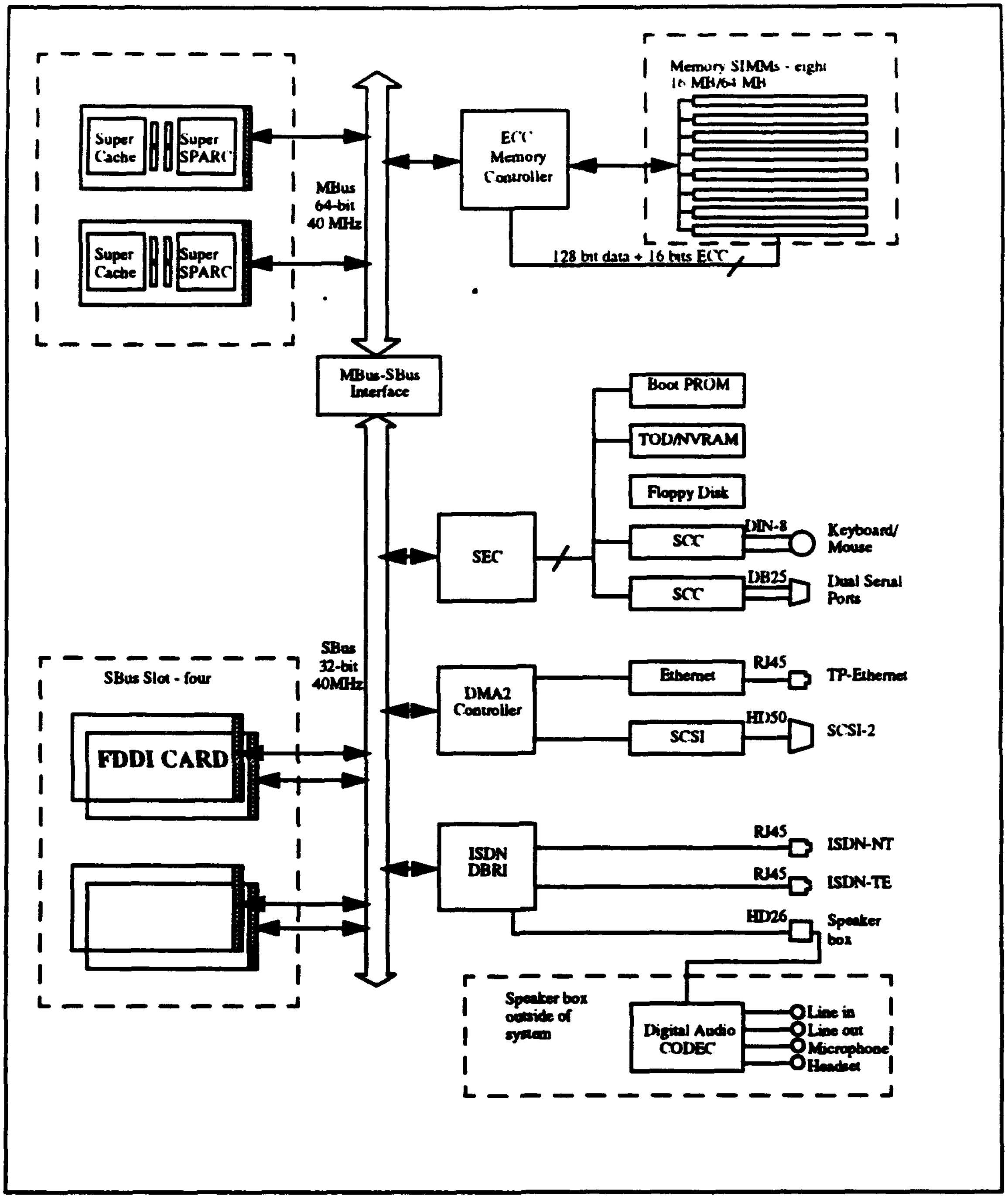

Figure 11: Sun-4m Architecture Used in the SPARCstation 10 System

I/O architecture: A single Application-Specific Integrated Circuit (ASIC) serves as the interface between the MBus and the SBus. The MBus is used as the processor memory interconnect, while the SBus is used only for I/O. The SPARCstation 10 system supports four SBus slots. They provide the means to interface a variety of I/O options, including network interfaces such as FDDI, graphics adapters and laser printer interfaces.

2. Silicon Graphics IRIS Indigo

The Silicon Graphics IRIS Indigo used in this test was an IRIS-4D^{TM}, model 4D/RPC. The IRIS Indigo uses the R3000A CPU RISC processor from MIPS Computer Systems Inc. It is assisted by a 32 Kbyte data and instruction cache and a MIPS R3010A floating-point unit. To speed up data transfers, IRIS Indigo uses custom ASICs designed by Silicon Graphics. These chips manage memory and processor interrupts, handle I/O and control the bus, often without CPU intervention [SILIC91].

We had one IRIS Indigo, Black, available for our FDDI research. This system has one 33 MHz processor, one 1 GB internal hard disk drive and 32 Mbytes of RAM. The workstation has the following features:

- A single 33 MHz chip with integer, floating point, memory management, and caches.
- 32-Kbyte instruction cache and 32-Kbyte data cache.
- Integral support for cache-coherent multiprocessing

a. Software Architecture.

The IRIS Indigo uses IRIX 4.0 which is Silicon Graphics' implementation of the UNIX operating system. IRIX 4.0 is based on AT&T UNIX System V.3, but also includes numerous 4.3 BSD extensions, such as TCP/IP network protocols and NFS, which provide transparent access to files across a heterogeneous network

b. Hardware Architecture.

This IRIS Indigo CPU board, Figure 12 [SILIC91], contains four functional sections:

- The processor core, which contains the CPU and FPU.
- Main memory, which contains DRAM and supporting circuitry
- The I/O system, which contains peripheral ports and hardware designed to read incoming data, manage incoming and outgoing data
- The audio system, which contains audio ports and digital signal processing hardware.

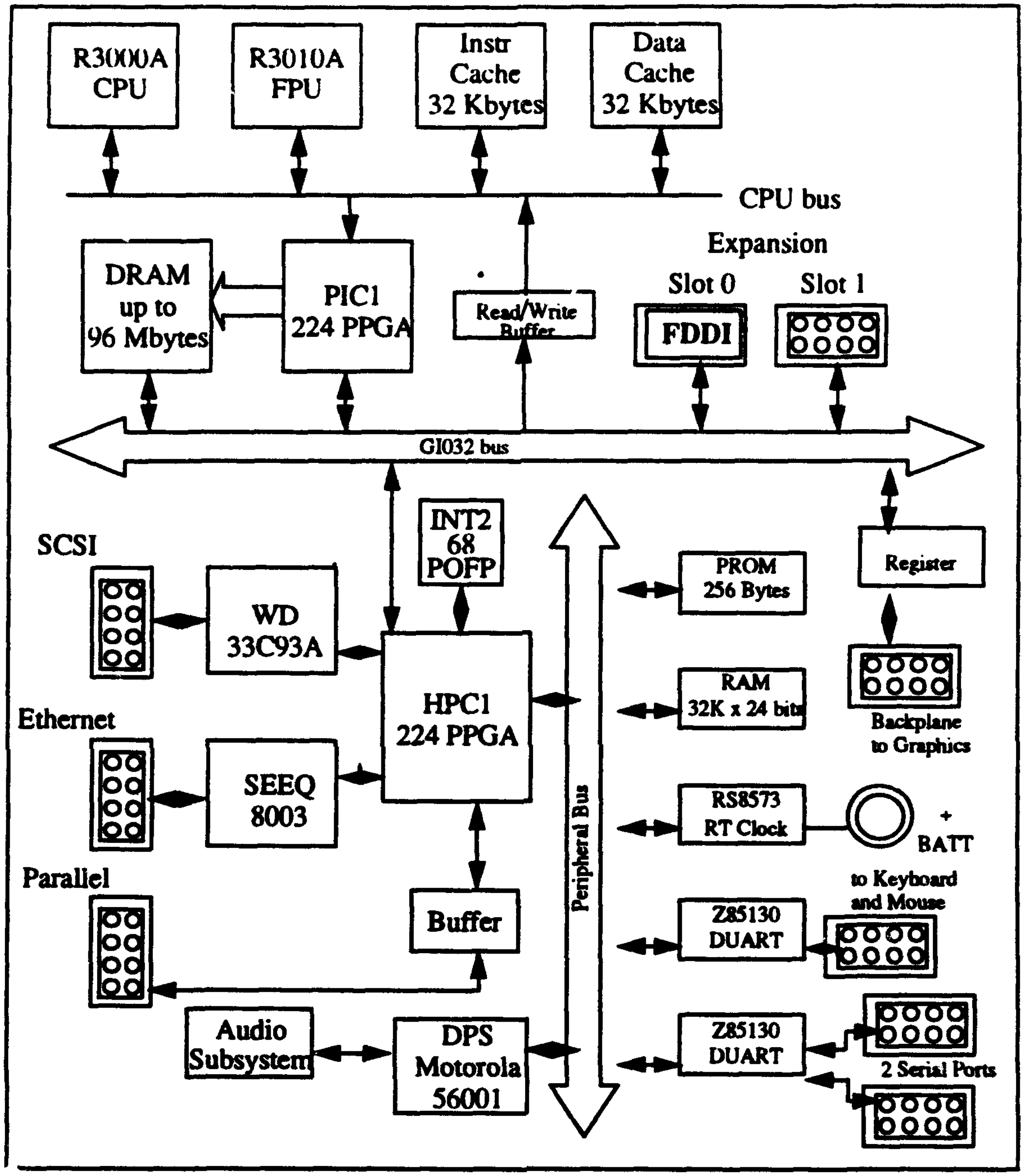

Figure 12: The IRIS Indigo CPU Board

Three busses connect parts of the CPU board:

- The CPU bus, which connects the CPU, FPU, cache control, and bus control hardware.
- The GIO32 bus, which is the main system bus connecting the processor core,

main memory, I/O system, expansion slots, and graphics board.

- The Peripheral bus, which connects the peripheral ports, audio system, and other I/O components.

The CPU bus and the GI032 bus have separate clocks and run at different speeds so that each part runs at maximum capability. The CPU and other chips can be upgraded independently as technology improves.

Instruction and Data Caches. Each cache is a 32 Kbyte cache.The instruction cache holds frequently used instructions and the data cache holds frequently used data. The IRIS Indigo uses a write-through scheme in the data cache to ensure that writes made to the cache are also written to the corresponding page in main memory.

The GI032 Bus. This bus is the IRIS Indigo's main system bus, and is designed for high speed data transfer. It connects the main systems of IRIS Indigo; the processor core, main memory, the I/O systems, the graphics system, and any systems plugged into the expansion slots.This bus is a synchronous, multiplexed address/data, burst mode bus that operates at 33.3 MHz, clocked independently of the CPU. The bus protocol supports data transfers at a maximum sustained rate of one word per clock.

The I/O System. The I/O system ties together a variety of I/O ports and the chips that drive them, a system clock, system Programmable Read-Only Memory (PROM) for booting up, an static RAM.

The HPC1 ASIC. The HPC1 is a custom Silicon Graphics chip that connects to the GI032 bus, the peripheral bus, and directly to several of the I/O ports. It is the heart of the I/O system, and quickly transfers data between main memory and a rich collection of peripheral devices.

Expansion Slots. The two expansion slots, connected directly to the GI032 bus, provide direct access to the system for Silicon Graphics and third party plug-in boards for such applications as high-speed networking, image compression, video deck control, and additional I/O. Slot 0 is used for our FDDI connection.

IV. TEST DESIGN PLAN

A. TEST STRATEGY

The objective is to find the upper limit of throughput by measuring actual throughput between high performance workstations over an FDDI network and to determine what bottlenecks, if any, exits between Sun Microsystem SPARC 10 multiprocessors running the Solaris 2.3 and NPI's FDDI network interface cards. This process will include identifying the various parameters which affect throughput and testing these parameters in enough detail to determine their impact on network performance. As explained in Chapter II, there are various levels of software that are involved in transferring data. As shown in Figure 13, as data is transferred from White to Gold, there are several impacts on the data transfer rate.

The key to this test design plan will be gathering the appropriate data to determine what impact these various parameters have on the transfer rate, and how to measure them. Three different methods will be used to measure the performance of data being transferred between workstations across the FDDI network. First, a commercial benchmarking tool will be used to provide performance results on the workstations. Second, a public domain networking benchmark tool will be used to show the transfer rate of the network. Third, a simple program which issues an *rcp* command and measures the time of the file transfer will be used.

B. NEAL NELSON BENCHMARK

The primary benchmarking tool to be used for providing the performance results on the workstations will be the Neal Nelson Business Benchmark™. This benchmark tool has been around for over 9 years and has been used as a tool for verifying vendor compliance during government contract awards. The Business Benchmark differs from other popular benchmarks in that its primary focus is not to provide a single number speed rating for a system, nor is its primary purpose to emulate a particular user group or duplicate the load created by certain task mix. The Business Benchmark was designed to incrementally stress various parts of a computer system and record how the system performs. The benchmark was intended to uncover both the strengths and the weaknesses of a computer architecture and report them separately so that they can be understood and analyzed [GRAY91].

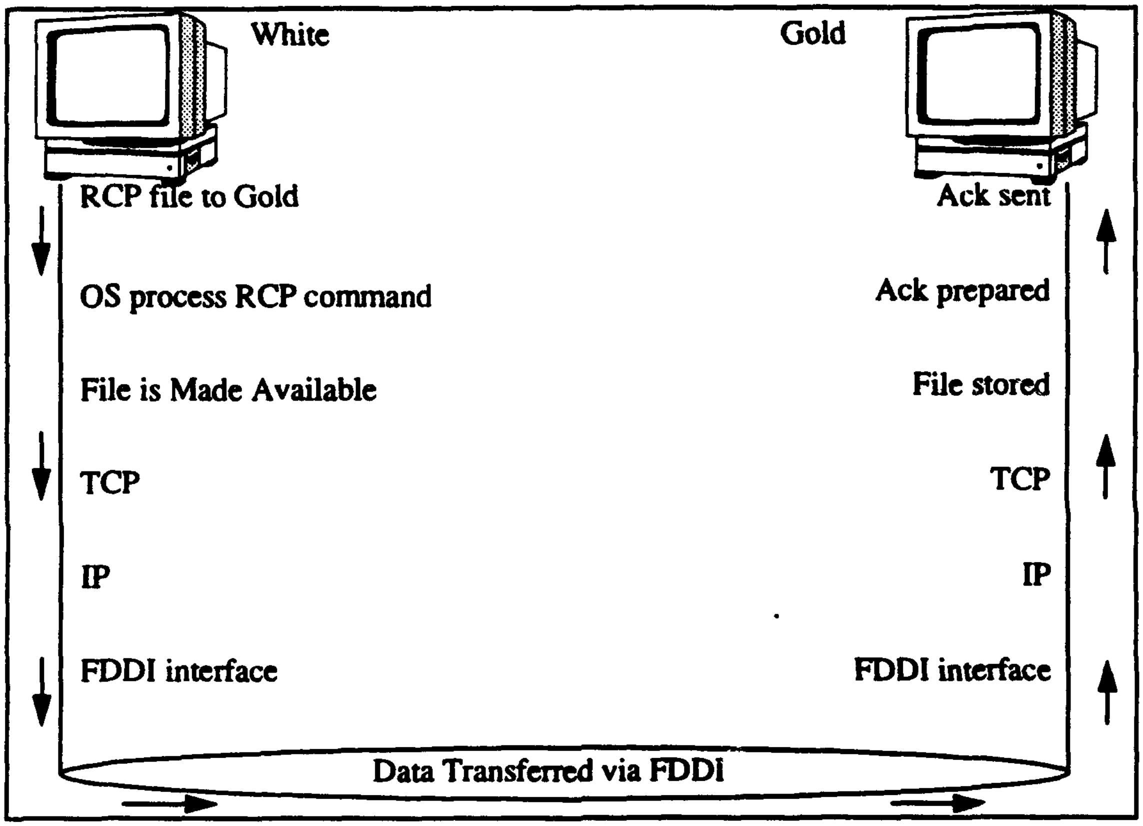

Figure 13: Flow of Data Across the FDDI Network Using the RCP Command

The Neal Nelson Business Benchmark is a multitasking benchmark with a parent/child design. A parent process creates child processes and instructs them to run tests in various combinations. There can be from one to one hundred child processes running simultaneously during a benchmark session. During a test session the parent process creates a single child process and instructs the child to perform a series of tests. Then the parent creates a second child and directs both children through the same series of tests. This process is repeated until a desired maximum number of child processes is reached, or until the system runs out of some resource such as disk space [NNBM94].

The benchmark consists of thirty tests, which are divided into three groups.

Group 1: Tests a of mix of activities that are intended to approximate the processing activities for the following five types of users. Group 1 includes the following tests:

1) Simulated Office Automation Workload
2) Simulated Database Workload
3) Simulated Software Development Workload
4) Simulated Transaction Processing Workload
5) Simulated Calculation Workload (Math/Statistics/CAD/CAM)

Group 2: Tests designed to perform various types of calculation tasks and thereby profile the performance of the computer's calculation subsystem. Group 2 includes the following tests:

6) Write to Shared Memory
7) Read from Memory, Small Instruction Area, Small Data Area
8) Read from Memory, Small Instruction Area, Larger Data Area
9) Read from Memory, Larger Instruction Area, Small Data Area
10) Read from Memory, Larger Instruction Area, Larger Data Area
11) Make Machine Page or Swap with 'malloc' and 'free'
12) Combined Integer and Floating Point Math
13) Math Library Functions
14) Semaphores, Shared Memory, Context Switch
15) Write to and Read from Pipes, Context Switch
16) Sample System Calls
17) Increasing Depth of Function Calls

Group 3: Tests that perform a series of disk input and output functions to profile the performance of the disk subsystem. Group 3 includes the following tests:

18) 1024 byte Sequential Reads from Unix File(s)
19) 1024 byte Sequential Writes from Unix File(s)

20) 8192 byte Sequential Reads from Unix Files(s)
21) 3192 byte Sequential Writes to Unix File(s)
22) 4096 byte Synchronized Reads from Unix File(s)
23) 4096 byte Synchronized Reads from Raw Device(s)
24) 16384 byte Synchronized Reads from Unix File(s)
25) 16384 byte Synchronized Reads from Raw Device(s)
26) 4096 byte Pseudo Random Reads from Unix File(s)
27) 4096 byte Pseudo Random Reads from Raw Device(s)
28) Profile Disk Cache for Unix File(s)
29) Profile Disk Cache for Raw Device(s)
30) 8192 byte Sequential Writes then 'sync'

During each of the above tests, measures will be obtained at load factors from 1 to 20. This load factor number indicates the number of copies of the benchmark program which were running simultaneously. Each load factor unit might approximate the workload of one or two heavy users or possibly twenty light users. The measurements will be in seconds to complete the measured task. The system which takes less time to accomplish the measured task is the faster system.

C. NEW TEST TRANSMISSION CONTROL PROTOCOL

New Test TCP *(nttcp)* uses Test TCP (*ttcp)* as the basic tool for determining measured throughput over any physical network media. *nttcp* provides the option of dynamically changing the TCP/IP window size during the throughput test. *ttcp* was developed by the U. S. Army's Ballistic Research Lab (BRL) which is now the U. S. Army's Research Lab (ARL) and is considered one of the default network performance benchmarks.

nttcp tests TCP and UDP performance by timing the transmission and reception of data between two systems using the UDP or TCP protocols. It differs from common "blast" tests, which tend to measure the remote *inetd* as much as the network performance, and which usually do not allow measurements at the remote end of a UDP transmission.

For testing, the transmitter should be started with -t after the receiver has been started with -r. For testing various window sizes, *nttcp* allows a -w option which permits the user to specify the desired TCP/IP window size. Some of the other options which were used during this investigation are shown below:

-t	Transmit mode.
-r	Receive mode.
-u	Use UDP instead of TCP.
-n	Number of source buffers transmitted.
-l	Length of buffers in bytes.
-w	TCP/IP window size in k bytes.
-p	Port number to send to or listen on.

Below are the commands used in a typical session during this investigation:

Receiving system (gold):
gold: *nttcp* -r -p3000 -w12
Transmitting system (white):
white: *nttcp* -t -p3000 -l65536 -n1024 -w12 gold

The shell scripts along with the *nttcp* program are in Appendix A. The shell scripts *doit.sh* and *ttest.sh* were written by personnel at the U. S. Army Research Lab (ARL) and modified to fit this investigation. These scripts were designed to be used with the program *nttcp*. The first script, *doit.sh*, provides the various combinations of data sizes to be transferred along with starting and stopping times of each run. This script runs through six iterations of identical data sets. The shell script *ttest.sh*, provides the calls to the program *nttcp*. Using the data length and number of packets specified in the shell script *doit.sh*, *ttest.sh* makes numerous calls to *nttcp* varying the window size from 4 k to 60 k in 8 k increments. This combination of amount of data transferred, number of test runs and number of window sizes provides a total of 576 measured data transfers during a single run. Amount of data transferred (12 sizes) * number of test runs (6 runs) * number of window sizes (8 different window sizes) = 576 measured data transfers. Below is an example of the results from a single call to *nttcp* with the amount of data to be transferred equal to 33,554,432 bytes of data and the TCP/IP window size being varied from 4 k to 60 k in 8 k increments:

Window Size(bytes)	Transfer Rate (Mb/s)
4096	32.7680
12288	29.1271
20480	37.4491
28672	43.6907
36864	52.4288
45056	43.6907
53248	43.6907
61440	37.4491

The TCP/IP window size is adjusted during these runs using the *setsockopt* system call. After the window size has been adjusted, the *getsockopt* system call is performed to verify that the TCP/IP window size has been changed as requested. Figure 14 shows an example of the *setsockopt* and *getsockopt* system calls used in the *nttcp* program.

```
if (setsockopt (fd, SOL_SOCKET, SO_SNDBUF, (char *) &sendwin, sizeof(sendwin)) < 0 )
        printf("get send window size didn't work\n");
if (setsockopt (fd, SOL_SOCKET, SO_RCVBUF, (char *) &rcvwin, sizeof(rcvwin)) < 0 )
        printf("get rcv window size didn't work\n");

if (getsockopt (fd, SOL_SOCKET, SO_RCVBUF, (char *) &sendwin, &optlen) < 0 )
        printf("get send window size didn't work\n");
else printf("send window size = %d\n", sendwin);

if (getsockopt (fd, SOL_SOCKET, SO_RCVBUF, (char *) &rcvwin, &optlen) < 0 )
        printf("get rcv window size didn't work\n");
else printf("receive window size = %d\n", rcvwin);
```

Figure 14: Example of *setsockopt* and *getsockopt* System Calls

D. REMOTE COPY PROTOCOL TRANSFER

Another program being used to measure the data transfer rate is a simple C program which issues a *rcp* command transferring a file from one workstation to another (Appendix B). The primary reason for choosing the *rcp* command is that it uses TCP which is a reliable transfer agent versus UDP which is unreliable. By using the *rcp* command, we are able to measure the time from the *rcp* command being issued to the time the *ack* is received back from the other workstation. The system can access the clock prior to issuing the *rcp* command, and then again after it receives the *ack* from the other workstation. Since the *rcp* provides for reliable data transfer, this allows a measurement of the total transfer time. Figure 15 shows the code obtaining the current system time, issuing the *rcp* command and then obtaining the system time again after the transfer is complete.

```
a = gettimeofday(&timestart, zonestart);
  if (a != 0)
    printf ("Oops ! %d\n", a);

  /* Use system call to do file transfer */
  system ("rcp large_file gold-fddi:/usr/test/gtow_test");

  /* Get stop time in sec&usec and check if successful */

  b = gettimeofday(&timedone, zonedone);
  if (b != 0)
    printf ("Oops! %d\n", b);
```

Figure 15: Implementation of RCP System Call

This method includes all the overhead from the operating system, *rcp*, TCP, IP and FDDI. After the *rcp* command is issued, the file is located in the file system and loaded into memory. Next, the workstation from which the command is being executed must perform a name/address resolution to determine where the file is being transferred. DNS provides this name/address resolution. Once this name/address resolution is performed the file is handed off to TCP to begin the transfer from workstation A to workstation B. TCP hands the file transfer off to IP which forwards the file to the FDDI protocol. At this point the FDDI SBus card transfers the file from workstation A to workstation B. At workstation B the reverse scenario takes place. The file is handed off from the FDDI protocol to the IP protocol, to the TCP protocol, and finally reaches the OS on workstation B. At this point, TCP on workstation B must issue an *ack* to let workstation A know that the file has been correctly received and handed off to the OS.

The *rcp* command copies files between machines. Each filename or directory argument is either a remote file name of the form:

hostname:path

or a local file name (containing no: characters, or a / before any: characters).

If a filename is not a full path name, it is interpreted relative to the users home directory on hostname. A path on a remote host may be quoted (using \, ", or ') so that the metacharacters are interpreted remotely.

rcp does not prompt for passwords; your current local user name must exist on hostname and allow remote command execution by *rsh*.

rcp handles third party copies, where neither source nor target files are on the current machine. Hostnames may also take the form

username@hostname:filename

To use username rather than your current local user name as the user name on the remote host. *rcp* also supports Internet domain addressing of the remote host, so that:

username@host.domain:filename

specifies the username to be used, the hostname, and the domain in which that host resides. Filenames that are not full path names will be interpreted relative to the home directory of the user named username, on the remote host.

E. PARAMETERS WHICH AFFECT BOTH TEST

The following driver parameters will be tuned under Solaris 2.3.

sbf_num_llc_rx	/* For LLC network traffic: /* Number of 4k receive buffers, maximum is 64 4k buffers /* Default is 48 4k buffers per NP-SB adapter
nfs_async_threads	/* Number of NFS thread for handling network file service /* Default is 8
sbf_treq	/* Amount of time for TTRT, default is 8ms /* Range is from 2ms to 165ms
sbf_mtu	/* Maximum protocol packet size, default is 4352 bytes

The above 4 tunable parameters along with the TCP/IP window size will be varied during the *rcp* and *nttcp* transfer test. The TCP/IP window size controls the amount of data permitted to be transferred between TCP acknowlegments. Numerous tests will be run varing each of the four parameters to determine what combination of values provides the optimum throughput performance and what weight each parameter has on the changes. The baseline test will be the values the manufacture recommends as the default values.

F. FILE SIZES FOR BOTH TRANSFERS

In order to measure the impact of the TCP, IP and FDDI overhead during the test, various sizes of files will be transferred. For the *rcp* test, the properties of the four files to be used are shown in TABLE 1. These files range in size from 6 bytes to 17,989,936 bytes. The amount of overhead during the transfers can be estimated as follows:

For the *nttcp* test, the amounts of data to be transferred is shown is TABLE 2. The amounts of data to be transferred is obtained by specifying the length of a buffer to be transferred and the number of buffers. As an example, if 2048 buffers of length 8192 bytes are transferred, then a total of 16,777,216 bytes of data are being transferred. The combinations listed in TABLE 2 give a range from 4,194,304 bytes to 2.684354e+08 bytes being transferred.

TABLE 1: RCP FILE SIZES AND ASSOCIATED OVERHEAD

File Size	Total Overhead
Huge (17,989,936 bytes)	1.37%
Large (1,314,923 bytes)	1.37%
Medium (48,072 bytes)	1.42%
Tiny (6 bytes)	90.9%

In order to make it easier to reference which file size has been used in the various test, the files will be referred to as File A through File H with File A being the smallest file, 4,194,304 bytes, and File H being the largest file, 268,435,400 bytes. The rest of the files are in order of size from the smallest file to the largest file.

TABLE 2: FILES (DATA SIZES) FOR NTTCP TEST

length of Buffers / Number of Buffers	8192 bytes (Files A - D)	65536 bytes (Files E -H)
512	4194304 bytes	33554432 bytes
1024	8388608 bytes	67108864 bytes
2048	16777216 bytes	1.342177e+08 bytes
4096	33554432 bytes	2.684354e+08 bytes

G. SYSTEM CONFIGURATIONS FOR ALL TESTS

As described in the previous sections, various tunable parameters and file sizes will be used during this investigation. In order to obtain reliable results, numerous test must be conducted to achieve a comfortable confidence level. Unfortunately, it is not practicable to perform all the test runs necessary to test all combinations possible let alone run enough iterations of each test to obtain the desired confidence level in the results.

As an example, just running the various combinations of tests described earlier with the *nttcp* program, there were 576 measured data transfers during a single run. One such test took a combined total of 3 hours and 15 minutes to run. During initial runs of the *nttcp* program, the TCP/IP window size was varied in 4 k increments. It was determined that there was little difference between the individual transfer rates of 4 k window sizes. Therefore, follow-on test were run at intervals of 8 k window sizes. This change reduced the run times from over 6 hours to just over 3 hours with little to no loss of usable results.

As noted earlier, there are other tunable parameters which can be modified by using the set command in the /etc/system file. Once again, it is not possible to test all possible combinations of parameters. As an example, if we start with the 576 measured data transfers which took over 6 hours with a 4 k TCP/IP window size increment, then test the TTRT parameter at 5 ms increments (33 tests), then the *sbf_num_llc_rx* buffers at 4 k increments (15 test), then the sbf_num_smt_rx buffers at 4 k increments (15 tests) and assume that we would like a confidence level which requires 50 runs of each test, we would have a total of 33*15*15*50 = 371250 tests needed to reach any conclusions. If each test took over six hours to conduct, it would take a total of 2,227,500 hours or 92,812.5 days just to finish conducting the tests.

In his book [JAIN91], Raj Jain discusses this dilemma of having too many variables to consider. The solution is to first get a gross picture of the impact of changing selective parameters. Once a parameter's impact on performance has been determined, then more thorough testing can be conducted by adjusting the correct parameters to obtain the desired confidence level. An example of this method in practice is changing from 4 k intervals in the TCP/IP window size to 8 k windows sizes.

In addition to the tunable parameters already discussed, this investigation is looking into the impact of the workstations running in multiprocessor modes and using a recently developed operating system, Solaris 2.3. This now doubles the required testing! First, tests will be conducted in the two processor configuration. Then, each Sun SPARCstation will be tested with only a single process, but still running Solaris. Once again, it is not possible to test all possible tunable parameters especially in both hardware configurations. Once a pattern has been established in the single processor configuration, follow-on tests in the multi-processor hardware configuration will be focused to limit the scope of tests to changing those parameters which produce the best results.

H. PARAMETER BASELINE

First, a baseline condition must be established before any changes are made to the system. This baseline will be with the following parameter values shown in TABLE 3. This table pertains more to the parameter settings in the *nttcp* and *rcp* test than the Neal Nelson Benchmark test. The first parameter, *NFS_asynch_threads*, has an impact on all three test. The other three parameters only impact the results of the *nttcp* and *rcp* test. No changes will be made to the workstations other than the changes to the tunable parameters listed below. Stored with the results of each *nttcp* and *rcp* test run is a README file with the below parameters and their values for that test.

While the below parameters are changed for the *nttcp* and *rcp* test, the TCP/IP window size will also be varied. The TCP/IP window size is not listed below in TABLE 3 as a tunable parameter. It is being treated differently due to the method it is varied during the test transfers. The *nttcp* program will be varying the TCP/IP window size during the test whereas the below listed tunable parameters must be changed by rebooting the workstations in-between the various tests.

TABLE 3: DEFAULT PARAMETERS USED FOR ALL THREE TEST

	NFS_asynch_threads	t_req	sbf_num_llc_rx	sbf_mtu
Neal Nelson Benchmark	8	8ms	48K	4352
NTTCP	8	8ms	48K	4352
RCP	8	8ms	48K	4352

Below is a review of the parameter descriptions:

sbf_num_llc_rx	/* For LLC network traffic. Number of 4k receive buffers /* maximum is 64 4k buffers
sbf_mtu	/* Maximum protocol packet size, default is 4352 bytes
t_req	/* Token holding time, default is 8ms
nfs_asynch_threads	/* For NFS service. Number of threads alloted. Default is 8

The results of the initial *nncp* baseline test during the single processor test are shown below in TABLE 4. The results shown in this table are the averaged results obtained from running this test for six runs. The first column shows the TCP/IP window size used during the test. The next 8 columns which are labeled File A through File H, show the averaged measured throughput in Mbps achieved during this test run.

TABLE 4: TEST RESULTS IN SINGLE PROCESSOR MODE

Window Size (K bytes)	File A Mbps	File B Mbps	File C Mbps	File D Mbps	File E Mbps	File F Mbps	File G Mbps	File H Mbps
4	32.77	38.23	40.05	36.06	32.46	31.92	31.51	31.96
12	118.33	29.13	32.77	30.95	24.63	25.42	24.93	26.21
20	32.77	43.69	41.87	40.57	40.57	40.33	40.62	39.86
28	32.77	49.15	38.23	42.65	40.57	40.89	41.67	41.81
36	32.77	43.69	38.23	43.69	41.61	40.89	41.67	42.38
44	32.77	49.15	38.23	42.65	40.57	39.43	42.26	42.09
52	76.96	49.15	38.23	41.61	38.75	37.93	39.35	36.15
60	32.77	43.69	38.23	41.61	33.72	34.37	30.09	30.60

V. TEST RESULTS AND ANALYSIS

In this chapter, the results from the three tests discussed in Chapter IV will be presented. First, the results from the Neal Nelson Benchmark tests will be presented. These results will show that the newer, faster 50MHz processors should outperform the older 40MHz processors. Next, the results from the New Test TCP *(nttcp)* network throughput tests will be presented. These results will show under what conditions the highest throughput can be achieved and what throughput bottlenecks exists. Last, the results from the *rcp* transfer tests will be presented. These results will help to identify bottlenecks within the workstation as a whole. The *nttcp* tests directly access the TCP/IP layer and do not provide a true measure of all the overhead present in distributed processing.

A. NEAL NELSON BENCHMARK

The Neal Nelson Benchmark is the tool being used to measure the capabilities of the workstations and the operating systems being tested. It is important to verify that the hardware we believe will perform faster has been verified to perform faster.

To begin with, two system disks were configured with the Solaris 2.3 operating system and one system disk was configured with the SunOS 4.1.3 operating system. A three gigabyte disk was partitioned and half of it made into a Unix file system, leaving the other half as a raw disk partition. The source code for the benchmark was obtained, installed, and compiled under Solaris 2.3 and SunOS 4.1.3 with the default tuning parameters.

The benchmark was started in the background and took approximately 20 hours to run under each of the following four hardware configurations: Gold with two 50HMz processors and White with two 40MHz processors, each running Solaris 2.3; Gold with one 50MHz processor running Solaris 2.3; Gold with one 50MHz processor running SunOS 4.1.3. Solaris 2.3 is Sun Microsystem's new operating system based on AT&T System V unix while SunOS 4.1.3 is based on Berkley's unix.

Once the benchmark testing was completed, the results were collected and electronically mailed to Neal Nelson & Associates, where the test reports were generated. The results from the three different configurations discussed below are listed in Appendix C with approval from Neal Nelson & Associates.

1. Gold Versus White, Two Processors and Solaris 2.3

In group 1 tests, which are intended to approximate the processing activities of five types of users, Gold consistently performed the tasks approximately 20 percent faster than White.

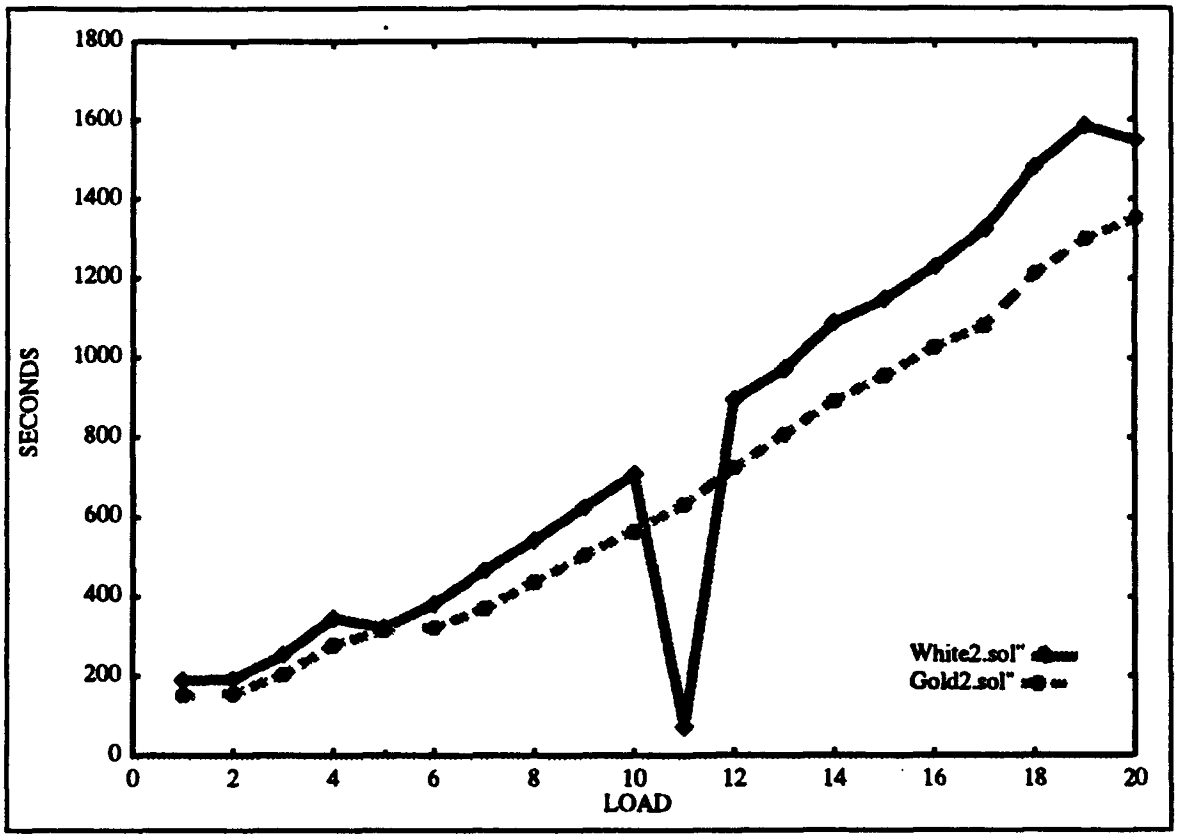

Figure 16: Gold Versus White, Two Processors

In group 2 tests, which are designed to perform various types of calculation tasks and thereby profile the performance of the computer's calculation subsystem, Gold continued to perform the tasks approximately 20 percent faster than White.

In group 3 tests, which performed a series of disk input and output functions to profile the performance of the disk subsystem, the results were mixed, but Gold still outperformed White on the average. These results varied from Gold outperforming White an average of 20 percent, to times when White outperformed Gold.

In Figure 16 on page 42 are the graphical results of Test 1, Simulated Office Automation Workload. Gold, with two 50MHz processors running Solaris 2.3, clearly took less time to perform the test than White with two 40MHz processors running Solaris 2.3 except at a load of 11. Once again, a load can signify either several light users or a single heavy user. As the loads increase you have either more light users or multiple heavy users.

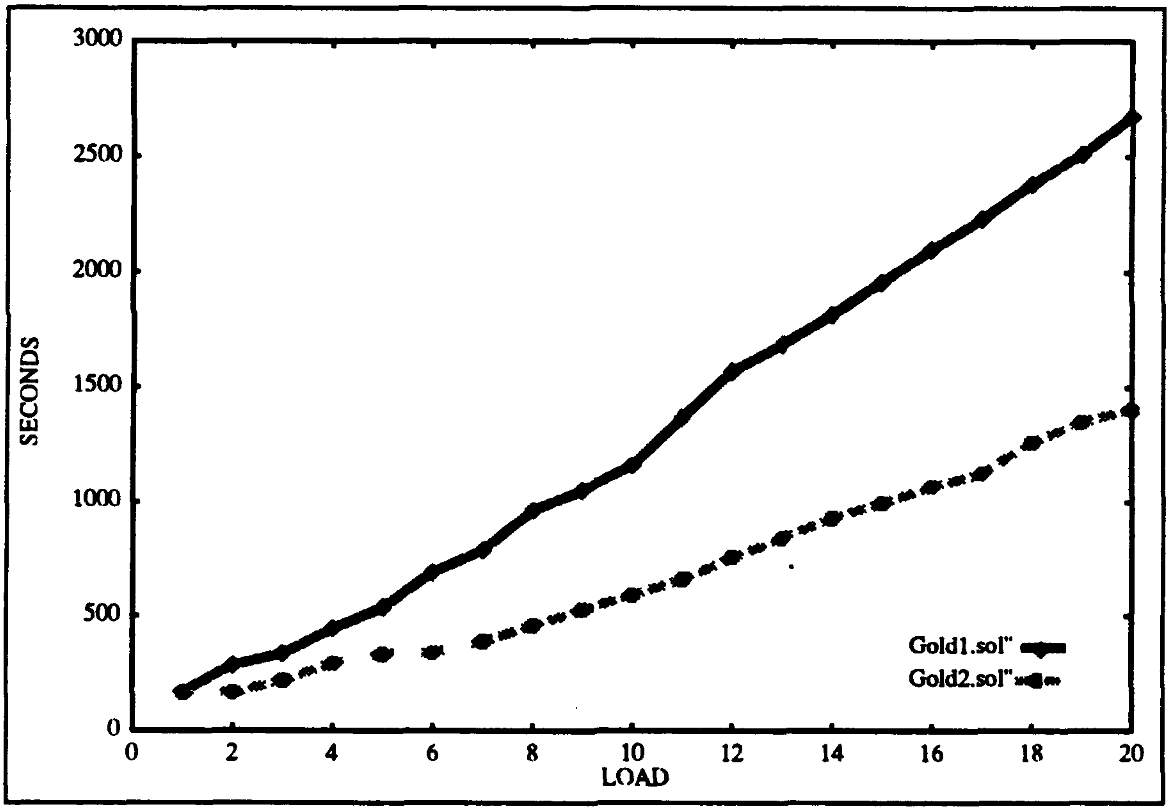

Figure 17: Gold One Processor Versus Gold Two Processors

2. Gold One Processor Versus Gold Two Processors and Solaris 2.3

In group 1 tests, the two processor configuration consistently outperformed the single processor configuration by 80 to 90 percent.

In group 2 tests, the two processor configuration continued to outperform the single processor configuration by 80 to 90 percent in all areas but one. In test 14, Semaphores, Shared Memory and Context Switch, the two processor configuration only outperformed the single processor configuration by 5 to 7 percent.

In group 3 tests, the results were once again mixed. The two processor configuration outperformed the single processor configuration in all tests but three by 50 percent. In test 19, 1024 byte Sequential Writes from Unix File(s) and test 21, 3192 byte Sequential Writes to Unix File(s), the single processor outperformed the two processor configuration by an average of over 200 percent. In test 30, 8192 byte Sequential Writes then 'sync', the single processor configuration outperformed the two processor configuration by approximately 20 percent.

In Figure 17 on page 43 are the graphical results of Test 1, Simulated Office Automation Workload. Gold with one 50MHz processor running Solaris 2.3 clearly took more time to perform the test than Gold with two 50MHz processors running Solaris 2.3.

3. Gold With One Processor, Solaris 2.3 Versus SunOS 4.1.3

In group 1 tests, the results were once again varied. SunOS 4.1.3 outperformed Solaris 2.3 in 4 of the 5 tests at the higher load levels by 3 to 4 percent. Solaris 2.3 outperformed SunOS 4.1.3 in two of the test at the lighter load levels by 3 to 4 percent.

In group 2 test, the results were more consistently in favor of SunOS 4.1.3. In 7 of the 12 test, SunOS 4.1.3 outperformed Solaris 2.3 by 4 to 5 percent. In test 13, Math Library Functions, SunOS 4.1.3 outperformed Solaris 2.3 by an average of 40 percent. Solaris 2.3 only outperformed SunOS 4.1.3 in three of the test areas. Two of the areas the percent was once again, only by 2 to 3 percent. In test 17, Increasing Depth of Function Calls, Solaris 2.3 outperformed SunOS 4.1.3 by an average of 40 to 50 percent.

In group 3 tests, the results were once again varied. In 6 of the tests, SunOS 4.1.3 outperformed Solaris 2.3 by anywhere from 15 to over 500 percent. In seven of the tests, Solaris 2.3 outperformed SunOS by anywhere from 100 to over 400 percent. Once again though, it appears that SunOS 4.1.3 came out slightly ahead in the high load area over Solaris 2.3

Below in Figure 18 are the graphical results of Test 1, Simulated Office Automation Workload. Gold with one 50MHz processor running SunOS 4.1.3 slightly beat out Gold with one 50MHz processor running Solaris 2.3 at the higher loads.

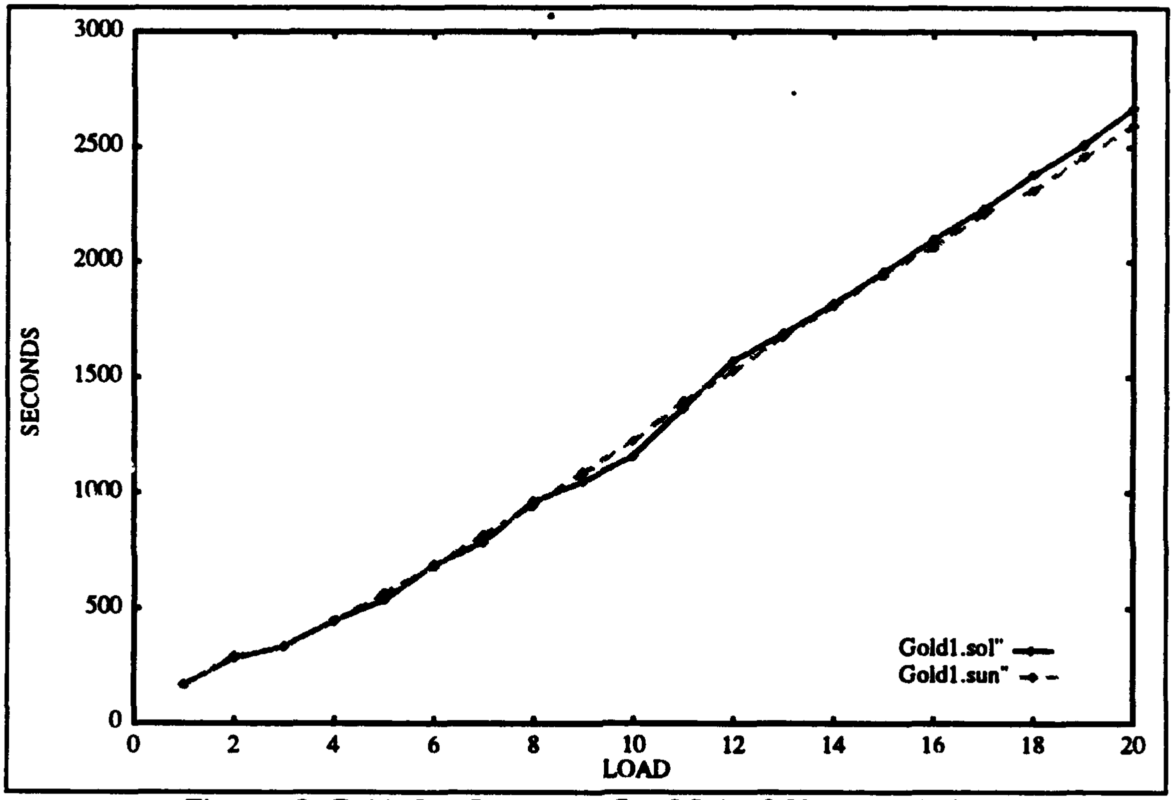

Figure 18: Gold, One Processor, SunOS 4.1.3 Versus Solaris 2.3

B. NEW TEST TRANSMISSION CONTROL PROTOCOL

As discussed in Chapter IV, the file sizes used during the test runs with New Test TCP *(nttcp)* are shown below in TABLE 5. The files are created by specifying the length of the buffer to be created and the number of buffers to be sent. The files will be referred to as File A through File H with File A being the smallest file, 4,194,304 bytes, and File H being the largest file, 268,435,400 bytes. The rest of the files are in order of size from the smallest file to the largest file.

TABLE 5: FILES (DATA SIZES) FOR NTTCP TEST

length of Buffers · Number of Buffers	8192 bytes (Files A - D)	65536 bytes (Files E -H)
512	FILE A 4194304 bytes	FILE E 33554432 bytes
1024	FILE B 8388608 bytes	FILE F 67108864 bytes
2048	FILE C 16777216 bytes	FILE G 1.342177e+08 bytes
4096	FILE D 33554432 bytes	FILE H 2.684354e+08 bytes

After conducting several test runs and observing the results, it became obvious that some smaller file sizes were not large enough to obtain accurate results. Whenever data is transferred using the *nttcp* program, the actual CPU time is the time used for calculating the throughput. If the CPU time used is too small, less than 0.1 seconds, the results become unreliable. An example of an unreliable transfer rate is given below in Figure 19. The reason for the inaccurate throughput result is the small amount of CPU time taken during this data transfer.

Transfers using the number of buffers = 512 and the length of buffer = 8192 were the only ones which had the unreliable transfer rates. There were typically only one or two transfer rates in each test which were unreliable. However, the window size was not always the same at which the unreliable transfer rate occurred. Therefore, the results of File A transfers were not used in this analysis.

```
ttcp-r: nbuf=512, buflen=8192, port=2001
send window size = 12288
receive window size = 12288
ttcp-r: 4194304 bytes in 0.06 real seconds = 68266.67 KB/sec = 546.1333 Mb/s
```

Figure 19: *NTTCP* Output for File Size of 4194304 Bytes

1. Single Processor Results

The first 32 test were run while Gold and White were set up in a single-processor configuration running Solaris 2.3. These 32 test represent a small subset of all possible tunable parameter combinations. The primary focus of this first set of test was to determine the effect of modifying the TCP/IP window size, the *nfs_async_threads* and the *t_req* parameters. Additionally, tests were conducted transferring data from White to Gold, Gold to White and both ways simultaneously. The 32 tests and the values of the tunable parameters are listed in TABLE 36, Appendix D.

The data gathered in the above 32 tests was analyzed using multiple linear regression analysis according to the model $y = \beta_o + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_m x_m + \varepsilon$ which relates the behavior of a dependent variable y to a linear function of the set of independent variables $x_1, x_2, \ldots x_m$. The β_j's are the parameters that specify the nature of the relationship, and ε is the random error term. The dependent variable y in this model is throughput. Refer to Figure 20 on page 49 under the bold face number 12 for the list of β_j's used in this model.

The tool used to produce the multiple linear regression analysis is Statistical Analysis System (SAS). The SAS tool is used to assist data analysts in analyzing data using regression analysis. Below in Figure 20 is an analysis of data throughput between White and Gold in the single processor configuration using the results from tests 1 - 32. Below is a description of the output from SAS as explained in [SASI91]. The bold face numbers have been added to aid in a description of the output.

1. The name of the dependent variable is THRUPUT.

2. The degrees of freedom (DF) associated with the sums of squares (SS).

3. The Regression SS (called Model SS) is 61279.61308, and the Residual SS (called ERROR SS) is 65217.01718. The sum of these two sums of squares is the C TOTAL (corrected total) SS = 126496.63026. This illustrates the basic identity in regression analysis that TOTAL SS = MODEL SS + ERROR SS. Usually, a good model results in the MODEL SS being a large fraction of the C TOTAL SS.

4. The corresponding Mean Squares are the Sum of Squares divided by the respective DF. The MS for ERROR (MSE) is an unbiased estimate of σ^2, provided the model is correctly specified.

5. The value of the F statistic, 239.470, is the ratio of the MODEL Mean Square divided by the ERROR Mean Square. It is used to test the hypothesis that all coefficients in the model, except the intercept, are 0. In this case, this hypothesis is:

$$Ho: \beta_1 = \beta_2 = \beta_3 = \beta_4 = \beta_5$$

6. The *p* value (Prob>F) of 0.0001 indicates that some of the β_s are not equal to 0.

7. Root MSE = 6.04621 is the square root of the ERROR MS and estimates the error standard deviation.

8. Dep Mean = 30.21891 is simply the average of the values of the variable THRUPUT over all observations in the data set.

9. C.V. = 20.00803 is the coefficient of variation expressed as a percentage. This measure of relative variation is the ratio of Root MSE to Dep Mean, multiplied by 100.

10. R-SQUARE = 0.4844 shows that a large portion of the variation in THRUPUT can be explained by variation in the independent variables in the model.

11. ADJ R-SQ is an alternative R-SQUARE and is an alternative to R-SQUARE that is adjusted for the number of parameters in the model according to the formula

$$\text{ADJ R-SQ} = 1 - (1 - \text{R-SQUARE})((n - 1)/(n - m - 1))$$

where n is the number of observations in the data set and m is the number of regression parameters in the model, excluding the intercept. This adjustment is used to overcome an objection to R-SQUARE as a measure of goodness of fit of the model. This objection stems from the fact that R-SQUARE can be driven to 1 simply by adding superfluous variables to the model with no real improvement in fit. This is not the case with ADJ R-SQ, which tends to stabilize to a certain value when an adequate set of variables is include in the model.

Mode: SINGLE PROCESSOR MODEL
Dependent Variable: THRUPUT

1

Analysis of Variance

Source	2 DF	3 Sum of Squares	4 Mean Square	5 F Value	6 Prob>F
Model	7	61279.61308	8754.23044	239.470	0.0001
Error	1784	65217.01718	36.55662		
C Total	1791	126496.63026			

7	Root MSE	6.0462	10 R-square	0.4844
8	Dep Mean	30.21891	11 Adj R-sq	0.4824
9	C.V.	20.00803		

Parameter Estimates

12 Variable	DF	13 Parameter Estimate	14 Standard Error	15 T for H0: Parameter=0	16 Prob>|T|
INTERCEP	1	27.673306	0.68625789	40.325	0.0001
SINGLE	1	8.620893	0.28565645	30.179	0.0001
WHITRAN	1	5.140603	0.28565645	17.996	0.0001
NUMBUFF	1	-0.000246	0.00010718	-2.295	0.0219
LENBUFF	1	-0.000107	0.00000511	-20.927	0.0001
WINDSIZE	1	0.008507	0.00779192	1.092	0.2751
TTRT	1	0.016060	0.01864409	0.861	0.3891
THREADS	1	0.008069	0.03570706	0.226	0.8212

Figure 20: SAS Analysis of Single Processor Transfers

12. The labels INTERCEP, SINGLE, WHITRAN, NUMBUFF, LENBUFF, WINDSIZE, TTRT and THREADS identify the coefficient estimates. The parameter SINGLE is used to show if the transfers were just between one workstation at a time, or if both White and Gold were transmitting at the same time. The parameter WHITRAN is used to show if White is transmitting or if Gold is transmitting. The other parameters were previously describ... .ı Chapter IV, Test Design Plan.

13. The Parameter Estimates give the fitted model

THRUPUT = 27.673306 + 8.620893(SINGLE) + 5.140603(WHITRAN)
- 0.000246(NUMBUFF) - 0.000107(LENBUFF)
+ 0.008507(WINDSIZE) + 0.016060(TTRT) + 0.008069(THREADS)

Thus, for example, a window size of 1k contributes 0.008507 to the throughput of data if all other parameters are held fixed. If the window size is 45k, then it contributes 0.382815 if all other parameters are held fixed.

14. These are the (estimated) standard errors of the parameter estimates and are useful for constructing confidence intervals for the parameters.

15. The *t* tests (T for H0: Parameter = 0) are used for testing hypotheses about individual parameters. The complete model for all of these *t* tests contains all the variables on the right side of the MODEL statement. The reduced model for a particular test contains all these variables except the one being tested. Thus, the t statistic = 0.008507(WINDSIZE) for testing the hypothesis Ho: $\beta = 0$ is actually testing whether the complete model containing NUMBUFF, LENBUFF, WINDSIZE, TTRT and THREADS fits better than the reduced model containing only NUMBUFF, LENBUFF, TTRT and THREADS.

16. The *p* value (Prob > |T|) for this test is $p = 0.0001$.

As shown in Figure 20 under item 16, Prob<|T|, the parameters NUMBUFF, WINDSIZE, TTRT and THREADS had the least impact on THRUPUT in this model. This shows up as the higher the Prob<|T| of the independent variable, the less impact it has on the dependent variable being modeled. Included in this model was the system transferring the data (WHITRAN) and whether it was a one way transfer or two way transfer (SINGLE). Therefore, the tunable parameters are competing with the fact that a 40MHz workstation is being compared to a 50MHz workstation and whether or not another station is competing for the token to transfer data.

The end result in this model is that the independent variable SINGLE has the most impact on THRUPUT and WHITRAN has the next largest impact on THRUPUT. This shows that competition for the token has more impact on throughput than tuning the system. However, there is still a performance gain to be realized with tuning the system for better throughput. In Figure 21 is a graphic comparison of the 1st Test with the 29th Test. As a reminder, the 1st Test is using the default parameters and the 29th Test is using the following parameter settings: *t_req* = 25ms; *nfs_async_threads* = 16; *sbf_num_llc_rx* = 48.

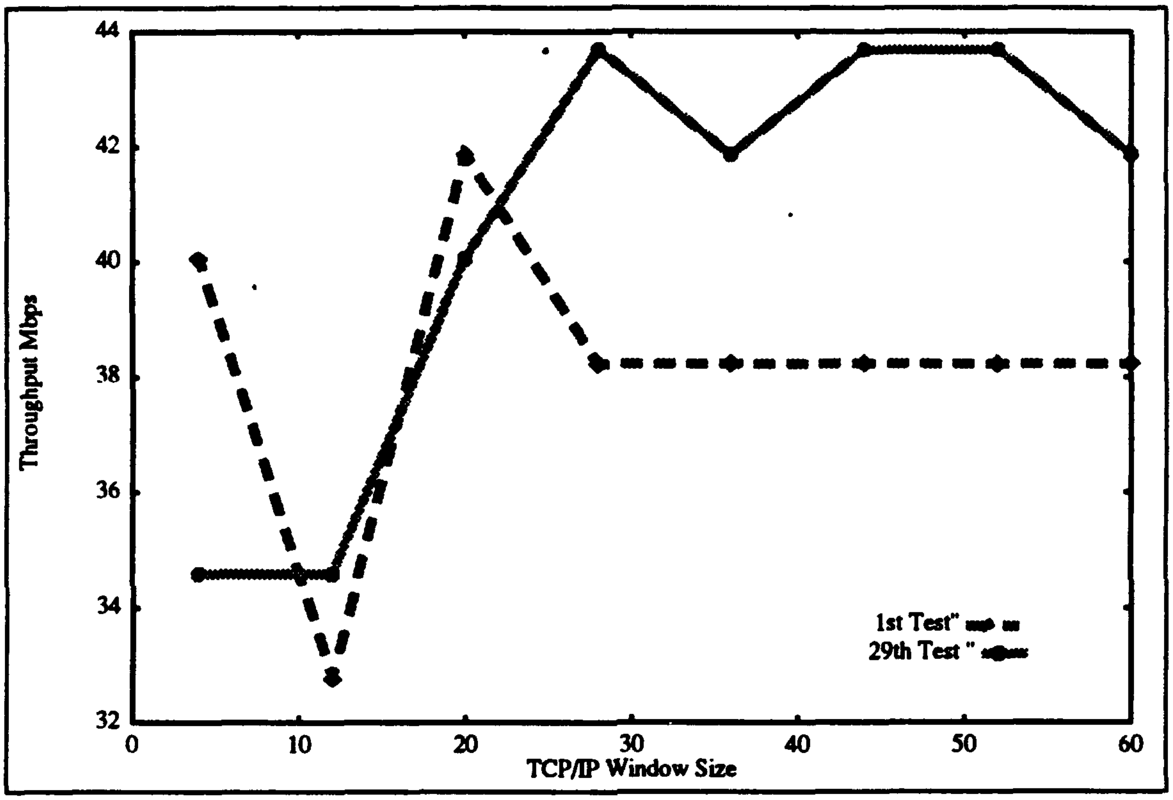

Figure 21: Single Processor, File D Transfer From White to Gold

2. Two Processor Results

The second set of test were run while Gold and White were set up in a two-processor configuration running Solaris 2.3. These 48 tests represent a small subset of all possible tunable parameter combinations. The primary focus of this set of test was to determine the effect of modifying the TCP/IP window size, the *nfs_async_threads*, *t_req*, *sbf_num_llc_rx* and the *sbf_mtu* parameters.The 48 test and the values of the tunable parameters are listed in TABLE 71, Appendix E.

The primary difference between this set of tests and the single processor test is that all transfers were made from White to Gold. To have also included transfers from Gold to White in this set of test would have doubled the number of transfers to 96 tests. Originally it was thought that by increasing the number of parameters being observed the R-square value would also have increased. The intention here was to account for more of the factors which impact the dependent variable THRUPUT.

Mode:TWO PROCESSOR MODEL
Dependent Variable: THRUPUT

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value	Prob>F
Model	7	66901.88212	9557.41173	68.151	0.0001
Error	2680	375842.31356	140.23967		
C Total	2687	442744.19568			

Root MSE	11.84228	R-square	0.1511
Dep Mean	40.72729	Adj R-sq	0.1489
C.V.	29.07702		

Parameter Estimates

Variable	DF	Parameter Estimate	Standard Error	T for H0: Parameter=0	Prob>|T|
INTERCEP	1	-91.980251	12.35679655	-7.444	0.0001
NUMBUFF	1	-0.000068737	0.00017141	-0.401	0.6884
LENBUFF	1	-0.000062619	0.00000817	-7.664	0.0001
WINDSIZE	1	-0.019754	0.01246095	-1.585	0.1130
TTRT	1	-0.024980	0.02981591	-0.838	0.4022
THREADS	1	-0.034226	0.05710325	-0.599	0.5490
LLC	1	0.643378	0.03496846	18.399	0.0001
MTU	1	0.024786	0.00285516	8.681	0.0001

Figure 22: SAS Analysis of Two Processor Transfers

As shown in Figure 22 on page 52, the R-square value decreased considerably between the single processor test and the dual processor test. As it will be shown later on, the cause for this decrease was the removal of the largest impact on throughput, competing with other stations for the token. Another indicator of the lack of confidence in the data being modeled is the large Standard Error for the independent variable INTERCEP. In the single processor model INTERCEP had a value of 0.68625789. In the dual processor model, the error has increased to 12.35679655.

The independent variables, NUMBUFF, THREADS and TTRT continued to have the least amount of impact on the dependent variable THRUPUT as indicated by their low Prob>|T| values. The independent variables with the largest impact were LENBUFF, LLC and MTU.

3. One And Two Processor Results

In the final analysis of both one and two processor tests, some additional facts need to be presented. There were a total of 4,480 throughputs measured in this analysis. There were 896 measurements in the one processor configuration and 2688 measurements in the two processor configuration. These are averaged measurements taken from the six runs in each 32 + 48 = 80 tests. Also, there were 896 measurements where both Gold and White were transmitting at the same time and 2688 measurements where only one station was transmitting.

When the model was first run including all the data from the one and two processor tests the R-square value was only 0.3559. This was higher than in the two processor model but lower than in the one processor model. A scatter plot was made of the various parameters to determine where there might be some problems with individual parameters. The most obvious problem was seen with the large variation of throughput with the parameter window size. At both the high end and the low end, the plot of window size versus throughput was not linear. By restricting the analysis of data to window sizes less than 50k and greater than 16k the R-square value increased to 0.6600. This reduced the number of measured observations from 4,480 throughputs to 2,240 measured throughputs.

Mode: ONE & TWO PROCESSOR MODEL
Dependent Variable: THRUPUT

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value	Prob>F
Model	10	179959.58511	17995.95851	432.681	0.0001
Error	2229	92708.03657	41.59176		
C Total	2239	272667.62168			

Root MSE	6.44917	R-square	0.6600
Dep Mean	42.53933	Adj R-sq	0.6585
C.V.	15.16048		

Parameter Estimates

Variable	DF	Parameter Estimate	Standard Error	T for H0: Parameter=0	Prob>|T|
INTERCEP	1	-70.427345	9.87019489	-7.135	0.0001
SINGLE	1	9.928996	0.43090313	23.042	0.0001
WHTRAN	1	3.652165	0.43090313	8.476	0.0001
NUMBUFF	1	-0.000052070	0.00010226	-0.509	0.6107
LENBUFF	1	-0.000047372	0.00000487	-9.719	0.0001
WINDSIZE	1	-0.200113	0.01523473	-13.135	0.0001
TTRT	1	-0.012831	0.01778717	-0.721	0.4708
THREADS	1	-0.039099	0.03406588	-1.148	0.2512
LLC	1	0.583336	0.02693145	21.660	0.0001
MTU	1	0.015782	0.00219894	7.177	0.0001
SD	1	9.535964	0.44849820	21.262	0.0001

Figure 23: SAS Analysis of Single and Two Processor Transfers

The results of the one and two processor analysis are above in Figure 23. One new independent variable, SD is used to model whether the transfer comes from the one processor tests or the two processor tests. Just as before, the independent variables

NUMBUFF, TTRT, and THREADS have the least amount of impact on THRUPUT. With the removal of the window sizes noted above, WINDSIZE now carries more weight in this model. The largest impact on THRUPUT in order of impact is caused by the variables SINGLE, SD, LLC and WINDSIZE. This statement will be covered in more detail later. This indicates once again that processor power has the largest impact on throughput. A graphical model of the difference is below in Figure 24. In this figure are plots of throughput from identical parameter configurations, but one is from a two processor run and the other is from a one processor run.

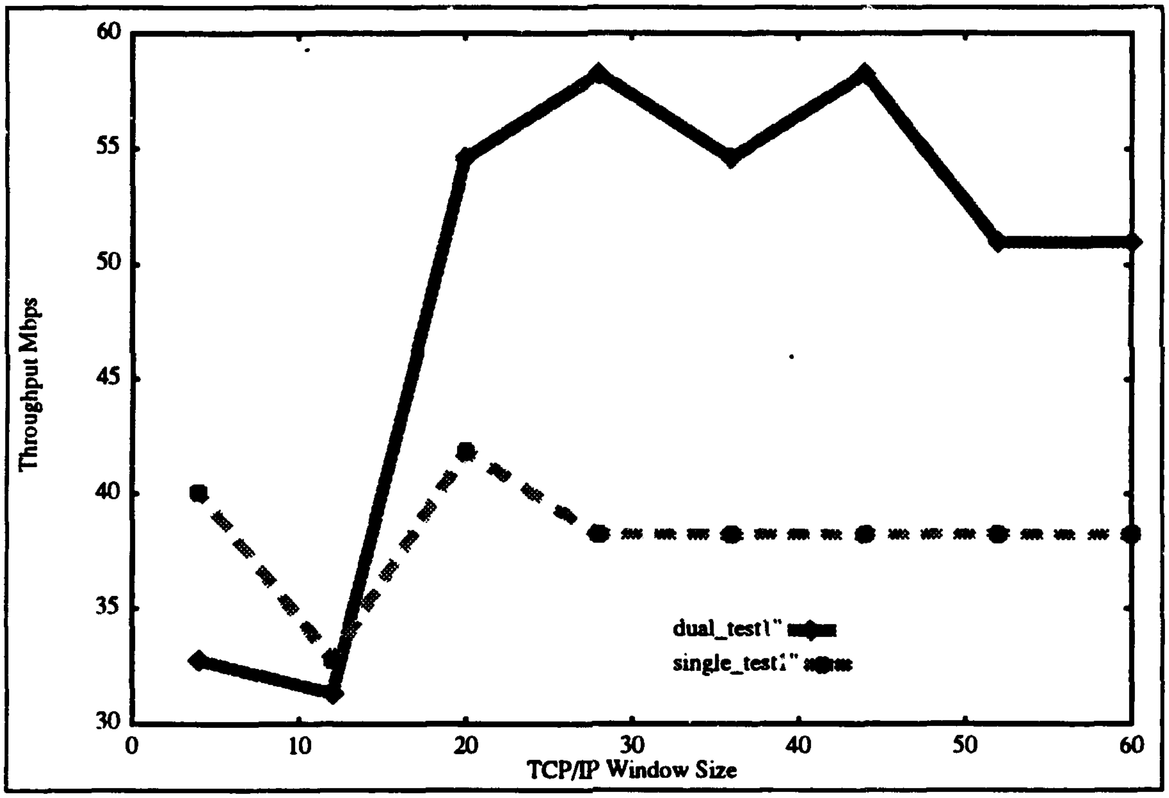

Figure 24: White Single Processor vrs White Two Processors

Another useful result which can be determined from the analysis of the one and two processor tests is a predicted throughput. Below in Figure 25 are SAS predictions of THRUPUT based on the 2,240 measured throughputs used in this analysis. To achieve the minimum predicted throughput, the following test was run using the parameter settings indicated in Figure 25. Data was transferred from Gold to White and White to Gold simultaneously. The results were taken from Gold with NUMBUFF = 4096, LENBUFF = 65536, WINDSIZE = 44, TTRT = 25, THREADS = 16, LLC = 40 and MTU = 4192. The results are below in TABLE 6.

The SAS predictions for the minimum predicted throughput was for a rate of 15.5302 Mbps. As shown in TABLE 6 the results from the actual tests was an average of 15.1463 Mbps and an mean of 15.0454 Mbps. Since the data used in the model was averaged data instead of mean data, the averaged achieved rate is the more accurate throughput rate to use. The SAS predictions for the maximum predicted throughput was for a rate of 58.7810 Mbps. As shown in TABLE 6 the results from the actual tests was an average of 60.07 Mbps and an mean of 65.5360 Mbps. In both cases the average throughput measured was very close to the predicted throughput. This shows that the SAS model was very accurate

OBS	SINGLE	WHITRAN	NUMBUFF	LENBUFF	WINDSIZE	TTRT	THREADS	LLC	MTU	SD	THRUPUT
1	1	1	1024	8192	20	5	8	56	4352	2	58.7544
2	0	0	4096	65536	44	25	16	40	4192	1	15.5302

Figure 25: SAS Throughput Prediction

TABLE 6: RESULTS OF SAS PREDICTIONS

PREDICTION	RUN1	RUN2	RUN3	RUN4	RUN5	RUN6	AVG	MEAN
LOW	17.4763	16.3840	12.8660	13.7069	20.1649	10.2802	15.1463	15.0454
HIGH	32.7680	65.5360	65.5360	65.5360	65.5360	65.5360	60.07	65.5360

The following formula relates the behavior of the dependent variable THRUPUT to a linear function of the set of independent variables SINGLE, WHITRAN, NUMBUFF, LENBUFF, WINDSIZE, TTRT, THREADS, LLC, MTU and SD. These are the values calculated in the One and Two Processor Model, Figure 23 on page 54.

$$\begin{aligned} \text{THRUPUT} = {} & -70.427345 + 9.928996(\text{SINGLE}) + 3.652165(\text{WHITRAN}) \\ & - 0.000052070(\text{NUMBUFF}) - 0.000047372(\text{LENBUFF}) \\ & - 0.200113\,(\text{WINDSIZE}) - 0.012831(\text{TTRT}) - 0.039099(\text{THREADS}) \\ & + 0.583336(\text{LLC}) + 0.015782(\text{MTU}) + 9.535964(\text{SD}) \end{aligned}$$

When the minimum and maximum throughput was predicted above in Figure 25 on page 56, it was simply a matter of inserting the largest parameter value in the above formula . the parameter estimate is positive and the smallest parameter value if the parameter estimate is negative. This resulted in the maximum predicted throughput. For the minimum predicted throughput, the largest parameter value is used if the parameter estimate is negative and the smallest parameter value if the parameter estimate is positive.

Below are the formulas for minimum and maximum throughput with the parameter estimates and parameter values multiplied together.

Maximum Throughput:

$$\begin{aligned} 58.7544 = {} & -70.427345 + 9.928996 + 3.652165 - 0.05331968 - 0.38807142 - 4.00226 \\ & - 0.064155 - 0.312792 + 32.666816 + 68.683264 + 19.071928 \end{aligned}$$

Minimum Throughput

$$\begin{aligned} 15.5302 = {} & -70.42734 + 0 + 0 - 0.21327872 - 3.1045714 - 8.804972 - 0.320775 \\ & - 0.625584 + 23.33344 + 66.158144 + 9.535964 \end{aligned}$$

Once the minimum and maximum throughputs were computed, the relative value of each parameter was calculated by subtracting the parameter's minimum value from it's maximum value. Below in Figure 26 are the results from this calculation. The value from the maximum calculation is listed, then the value from the minimum value is listed and finally the difference is listed. It is this difference which shows the impact each parameter has on the end throughput. The higher the difference is, the more weight that parameter carries in determining the maximum throughput.

	SINGLE	WHITRAN	NUMBUFF	LENBUFF	WIND.SIZE	TTRT	THREADS	LLC	MTU	SD
MAX:	9.92	3.65	-0.05	-0.38	-4.00	-0.06	-0.31	32.66	68.68	19.07
MIN:	0	0	-0.21	-3.10	-8.80	- 0.32	-0.62	23.33	66.15	9.53
DIFF:	9.92	3.65	0.16	2.72	4.8	0.26	0.31	9.33	2.53	9.54

Figure 26: Relative Importance of Each *nttcp* Parameter

The results listed above show that the following parameters, in order of importance, have the most impact on throughput using the current model:

- If the data was only being transferred from one workstation to another or if both workstations were transferring data to each other simultaneously.
- Whether the workstation had one or two processors
- The number of 4K receive buffers allotted for receiving data.
- The number of TCP/IP windows available for sending data.

Since the TCP/IP window size was limited in the above model to a range of 20k to 44k, this parameter showed up having less of an impact than it really has. As an example, in TABLE 72 on page 120 of Appendix E, the throughput rate for File C is 32.77 Mbps for a window size of 4k and 58.25 Mbps for a window size of 44k. That means the throughput rate at a 4k window size is only 56 percent the rate of the 44k window size. In this case, the window size has the largest impact on throughput performance. Unfortunately though, the results at the lower and higher window sizes were not consistent in all cases and the data was removed from the analysis. In most cases though, the difference in throughput performance between a TCP/IP window size of 4k and a window size of greater than 20k is more significant than any other factor considered in this investigation.

Based on the visual inspection of the results from both the one processor tests and the two processor tests, below is a revised list in order of importance the parameters having the most impact on throughput:

- The number of TCP/IP windows available for sending data.
- If the data was only being transferred from one workstation to another or if both workstations were transferring data to each other simultaneously.
- Whether the workstation had one or two processors
- The number of 4K receive buffers allotted for receiving data.

Another parameter which showed unexpected results is the WHITRAN parameter. This parameter is used to track any differences in throughput between transmitting data from White to Gold, or from Gold to White. The result in Figure 25 on page 56 indicates that transmitting data from White to Gold was faster than transmitting data from Gold to White. In the first 32 one processor tests, White had one 40MHz processor and Gold had one 50MHz processor. In the second 48 tests, White had two 40MHz processors and Gold had two 50MHz processors. Based on the Neal Nelson Benchmark tests, Gold should be capable of transferring data faster than White.

Several additional tests were conducted to determine why White was able to transmit data at a higher throughput than Gold. First, the FDDI cards were swapped to see if the FDDI card in Gold was causing the problem. The results of these tests are in TABLE 69 on page 117 and TABLE 70 on page 118. There was not any noticeable difference in throughput rates with the boards swapped. Next, the two 50MHz processors were placed in White and the two 40MHz processors were placed in Gold. The results of these tests are in TABLE 121 and TABLE 122 on page 137. As shown in Figure 27, even when both transmitting systems had two 50MHz processors and both receiving systems had 40MHz processors, White still had a higher throughput rate with File C than Gold.

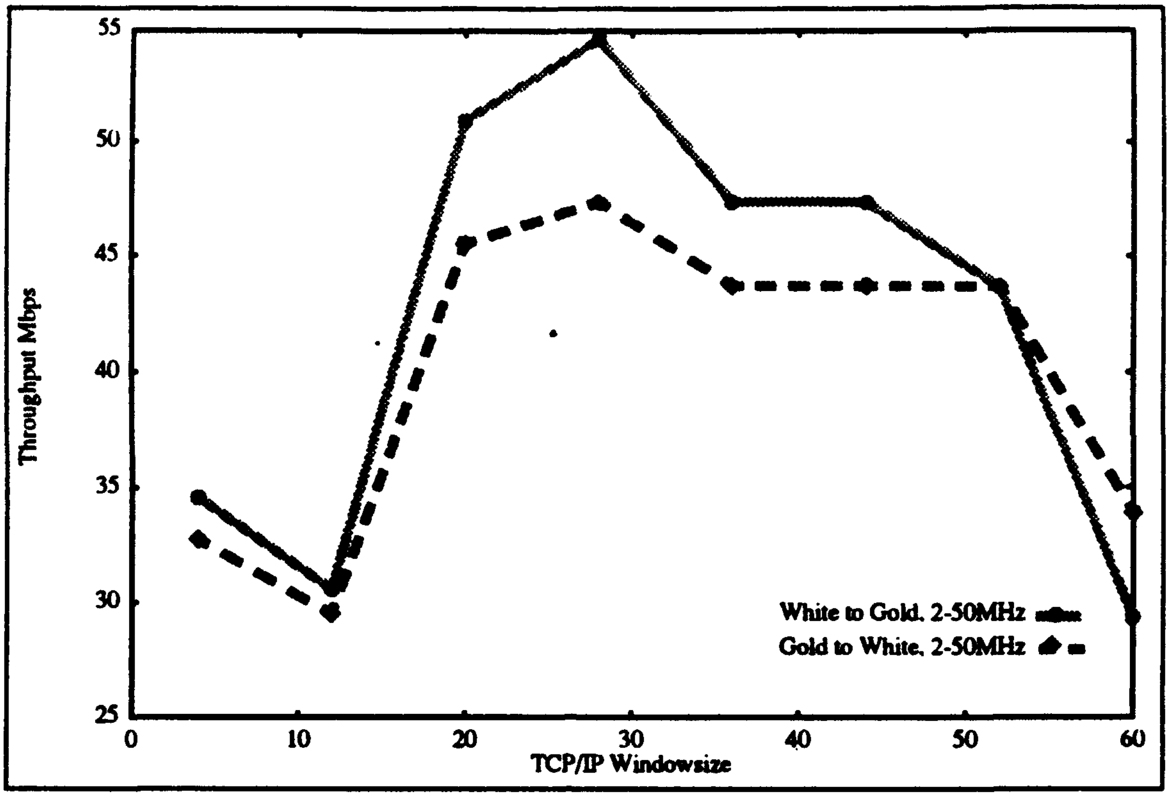

Figure 27: Throughput Comparison Between White and Gold

The only other difference between White and Gold is that Gold is the server on the FDDI network. Since the FDDI network only had three workstations on the network, this additional load on Gold should not be that great.

C. REMOTE COPY PROTOCOL TRANSFERS

Initially, the plan was to conduct file transfers using the *rcp* system call varying the tunable parameters just as in the *nttcp* tests. However, it was quickly observed that there were not any noticeable differences in measured throughput at the different parameter settings. This was understandable with the parameters *nfs_async_threads* and *t_req*. The SAS model showed that these tunable parameters had little effect on throughput. However, it was expected that there would be some different throughput rates with the TCP/IP window size, *llc* and *mtu* parameters varied.

The reason why the these parameters did not have an impact was that *rcp* does small size read()'s and write()'s, so the syscall overhead dominates over the time spent in the kernel in TCP. If an application wants optimum bulk data throughput, it should increase the receive buffering, and also do moderately large read()'s and write()'s so that the syscall overhead does not dominate. Also, *rcp* has to go through a complete login, exec of the user's shell, and run through the user's ".cshrc" or ".profile" on the server side before it begins transferring any data. If the data transfer is not really huge, the time spent logging in will be much greater than the time spent transferring the data.

Knowing that the largest impact on throughput based on the SAS modeled data is TCP/IP window size, processor power and whether or not another station is also transmitting, four different transfer tests were conducted with each of the four file sizes. As shown below in TABLE 7 and TABLE 8 on page 62, tests were conducted in the one processor configuration and the two processor configuration while transferring files one-way and two-way (between White and Gold simultaneously).

TABLE 7: RCP ONE PROCESSOR TRANSFER RESULTS

	TINY (6 bytes	MEDIUM (48,072 bytes)	LARGE (1,314,923 bytes)	HUGE (17,989,936 bytes)
ONE-WAY TRANSFER White to Gold				
TO: /FILE-NAME	.000032 Mbps	.25 Mbps	4.91 Mbps	13.20 Mbps
TO: /DEV/NULL	.000032 Mbps	.25 Mbps	5.85 Mbps	26.41 Mbps
TWO-WAY TRANSFERS White to Gold & Gold to White				
TO: /FILE-NAME	.000027 Mbps	.23 Mbps	4.47 Mbps	11.49 Mbps
TO: /DEV/NULL	.000027 Mbps	.22 Mbps	4.73 Mbps	16.72 Mbps

Also, files were transferred from disk to disk and from disk to /dev/null. This second transfer method does not result in a disk write at the destination workstation. The device driver, /dev/null, is used to dispose of files without needing to delete them. Files can be sent to /dev/null and this device driver accepts the data without writing them to disk.

The largest impact seen in this set of tests was the file size. The lowest throughput rate was observed when transferring the smallest file, TINY. This file has an associated overhead of 90.9% when being transferred over FDDI. The highest throughput was seen with the file HUGE. This file only had an overhead of 1.37% when transferred over FDDI. These overhead figures include the overhead associated with the FDDI, IP and TCP protocols. Another area with similar results as the *nttcp* test is whether the transfers are one-way or two-way. When the two workstations have to compete for the token the throughput drops.

TABLE 8: RCP TWO PROCESSOR TRANSFER RESULTS

	TINY (6 bytes	MEDIUM (48,072 bytes)	LARGE (1,314,923 bytes)	HUGE (17,989,936 bytes)
ONE-WAY TRANSFER White to Gold				
TO: /FILE-NAME	.000031 Mbps	.25 Mbps	4.94 Mbps	13.54 Mbps
TO: /DEV/NULL	.000031 Mbps	.25 Mbps	5.87 Mbps	28.42 Mbps
ONE-WAY TRANSFER Gold to White				
TO: /FILE-NAME	.000029 Mbps	.24 Mbps	5.26 Mbps	21.66 Mbps
TO: /DEV/NULL	.000029 Mbps	.24 Mbps	5.81 Mbps	29.82 Mbps
TWO-WAY TRANSFERS White to Gold & Gold to White				
TO: /FILE-NAME	.000029 Mbps	.24 Mbps	4.64 Mbps	13.27 Mbps
TO: /DEV/NULL	.000030 Mbps	.24 Mbps	5.55 Mbps	23.18 Mbps

The results during the *rcp* tests were much lower than during the *nttcp* tests. As an example, on the transfer of a file size of over 17 Mbytes from Gold with two processors to White:/dev/null, the best achieved throughput rate was 29.82 Mbps with *rcp*. This is only 29.82 percent of FDDI's available bandwidth and only 43.7 percent of the highest achieved throughput using *nttcp* (65Mbps). When transferring the same file from Gold to White and writing the file to disk, the transfer rate was 21.66 Mbps. This rate is only 72 percent of the transfer rate of transferring the data to /dev/null. Below in Figure 28 on page 63 is a graphical plot of the transfer rates just mentioned while transferring the 17.9 Mbyte file from Gold with two 50MHz processors to White with two 40 MHz processors.

There were two main differences between the transfer methods: First, the *rcp* transfers add another layer of protocols to the transfers. The *rcp* protocol hands off the data to be transferred to the TCP/IP protocol layers. This of course increases the amount of overhead transferred. Second, using *rcp* to transfer the data involves reading the data from disk before it can be transferred. Even though large amounts of data can be cached in the SuperCache 1-Mbyte external cache, this is not large enough for extremely large files being transferred to be completely cached. During this test files were transferred 9 times and then the median throughput rate was used for the results.

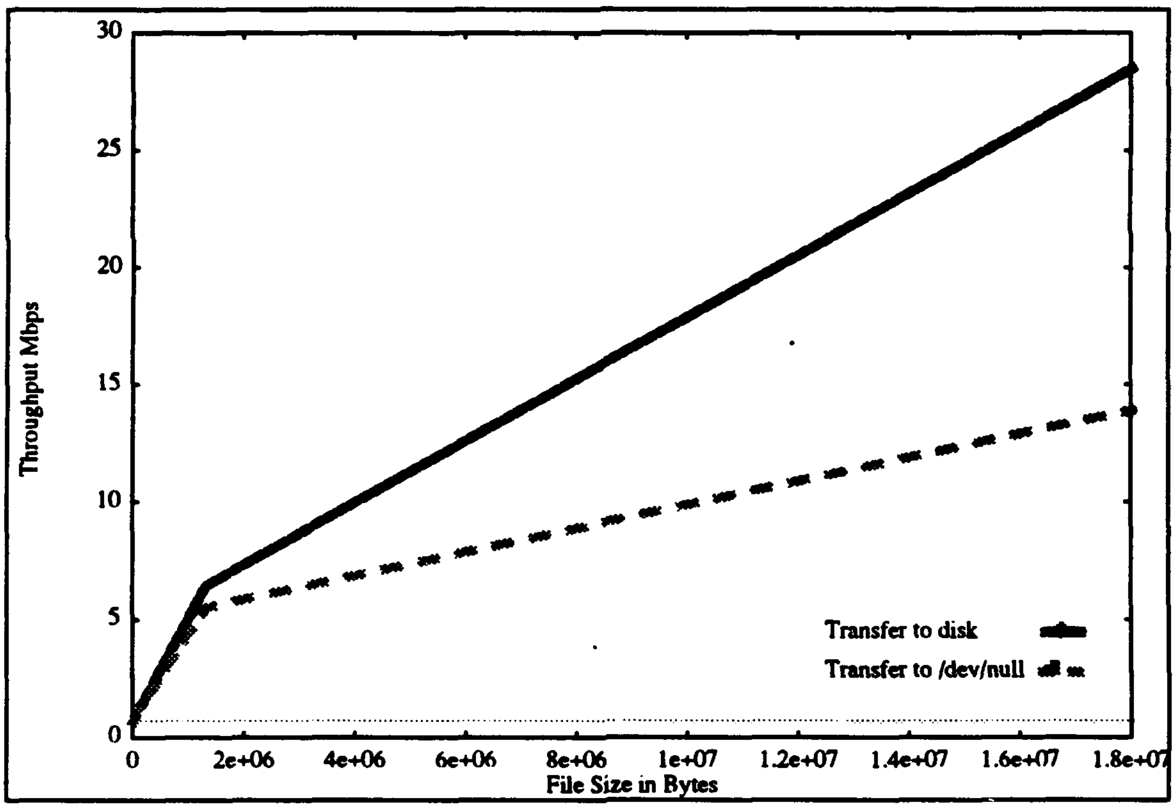

Figure 28: RCP File Transfers From Gold To White

The results from the *rcp* tests were pretty much as expected. The two processor transfers were faster than the single processor transfers and the one-way transfers were faster than the two-way transfers. However, the difference in these throughput rates was not as large as that seen with the *nttcp* tests. Since the additional overhead from the *rcp* system call should affect the transfer rates evenly, then the only other difference is that the data was transferred from disk instead of being generated by the CPU. The large difference in throughput rates achieved between the two test methods would indicate that the disk access is a very large bottle neck in throughput performance.

A quick comparison of the throughput rate observed using *nttcp* for a file size of 16,777,216 bytes (File C) and a *rcp* transfer of a file size 17,989,936 bytes shows a throughput rate of 32.77 Mbps for the *nttcp* transfer and a throughput rate of 28.42 Mbps when transferred to /dev/null. Both of these tests were one-way tests from White to Gold with both systems in the two processor configuration. In this comparison, the *rcp* tests had a throughput rate which is 86.7 percent of the *nttcp* throughput rate. This seems to indicate that the retrieval of the file from disk and the overhead of the *rcp* protocol are responsible for 13.7 percent of the slow down in throughput when transferring files.

When comparing the transfer rate of an *rcp* transfer from White to a file location on Gold with the *nttcp* throughput rate, there is a much larger difference in throughput. The *nttcp* throughput rate is still 32.77 Mbps and the throughput rate for the *rcp* file to file transfer is 13.54 Mbps. Here the *rcp* throughput rate is 41.3 percent of the *nttcp* throughput rate. This means that the time to receive and process the file at the destination workstation accounts for 45 percent of the reduced throughput. This is the 58.7 percent reduction minus the 13.7 percent attributed to the retrieval of the file from disk and the overhead of the rcp protocol.

D. ANALYSIS SUMMARY

The results from the Neal Nelson Benchmark showed that the systems being investigated were functioning as expected. The 50MHz system outperformed the 40MHz system and the two processor system outperformed the one processor system. One unexpected result was that SunOS 4.1.3 slightly outperformed Solaris 2.3 in just about every test except disk access to unix files. Solaris 2.3 was the clear winner in this area.

The *nttcp* results were analyzed using a linear multiple regression analysis model. Even though the throughput results were not linear, the model is believed to be accurate enough to show the relationship between the parameters being investigated. The analysis of this data provides the most concrete results of the two throughput tests methods.

The number of workstations on an FDDI network transmitting has the largest impact on throughput among the parameters investigated according to the one processor and two processor models. An example of this impact is to take the SAS prediction shown in Figure 25 on page 56 and change the parameter SINGLE from its one-way value to the two-way value. This allows SAS to predict a new throughput rate based on all the previous values except the change just noted. The result of the new prediction shows a new throughput prediction of 48.8254 Mbps. This is only 83.1 percent of the original throughput predication of 58.7544 Mbps.

The power of the workstation itself is a major factor in throughput potential. This is seen in the fact that the second largest impact on throughput in the one processor and two processor model is whether or not the workstation had two processors. The result of the new one processor prediction shows a throughput predication of 49.2184 Mbps. This is 83.7 percent of the original throughput predication of 58.7544 Mbps.

Since the TCP/IP window size was limited in the model to a range of 20k to 44k, this parameter showed up having less of an impact than it really has. In most cases though, the difference in throughput performance between a TCP/IP window size of 4k and a window size of greater than 20k is more significant than any other factor considered in this investigation.

The results from the *rcp* tests are more of an observation of the effects of the disk drive on throughput performance. Since both tests measure the time from start of test to receiving the *ack* from TCP on the receiving workstation that the data has been received, the only other real differences is the *rcp* protocol and the fact that the data is being transferred as files instead of being generated by the processor.

As pointed out earlier, the overhead of the *rcp* protocol and the time spent retrieving the file from disk is approximately 13.7 percent of the throughput rate observed during the *nttcp* throughput tests. Additionally, the overhead of processing the file at the receiving workstation is approximately 45 percent of the throughput rate observed during the *nttcp* throughput tests.

The observation made in the *nttcp* tests that white with only 40MHz processors could transfer data faster than Gold with 50MHz processors was not seen again in the *rcp* tests. In the *rcp* tests, Gold was able to transfer data at a higher throughput rate than White when Gold had the two 50MHz processors and White had the two 40MHz processors.

VI. CONCLUSIONS AND TOPICS FOR FUTURE RESEARCH

A. CONCLUSION

The objective of this research was to measure actual throughput between high performance workstations over an FDDI network to determine what bottlenecks, if any, exits between Sun Microsystem SPARC 10 multiprocessors running the Solaris 2.3 and Network Peripheral Inc.'s (NPI) FDDI network interface cards and to evaluate Transmission Control Protocol/Internet Protocol (TCP/IP) as a high speed transport protocol.

At the beginning of this investigation there were many speculations as to what throughput rates could be achieved and what effect varying the different tunable parameters would have on the throughput rates. It was assumed that the workstation with the 50MHz processor would have a faster throughput rate than the workstation with the 40MHz processors. It was also assumed that since Sun Microsystems was encouraging their users to switch from SunOS to Solaris, that Solaris 2.3 would clearly out perform SunOS 4.1.3.

The following sections outline the conclusions drawn from these investigations:

1. Workstation Conclusions

There were four benchmark tests conducted using the Neal Nelson Business Benchmark run on the two workstations, Gold and White.

- Gold had two 50MHz processors installed and was running Solaris 2.3.
- Gold had one 50MHz processor installed and was running Solaris 2.3.
- Gold had one 50MHz processor installed and was running SunOS 4.1.3.
- White had two 40MHz processors installed and was running Solaris 2.3.

Three test comparisons were conducted by Neal Nelson and Associates and the results can be summarized as follows:

- A workstation running Solaris 2.3 with two 50MHz processors can be expected to outperform a workstation running Solaris 2.3 with two 40MHz processors in most areas of performance by approximately 20 percent.

- A workstation running Solaris 2.3 with two 50MHz processors can be expected to outperform a workstation running Solaris 2.3 with one 50MHz processor in most areas of performance by approximately 90 percent.

- A workstation running SunOS 4.1.3 with one 50MHz processor can be expected to outperform a workstation running Solaris 2.3 with one 50MHz processor in most areas of performance by approximately 2 percent.

Of the three comparisons noted above, the first two results were expected. However, it was assumed that Sun Microsystem's release of Solaris 2.3 would result in improved operating system performance, not a slight drop in performance. These results were very important in the next step of the investigation. Knowing that the workstation with two 50MHz processors should outperform the workstation with two 40MHz processors helped isolate some unexpected results in workstation throughput.

2. Throughput Conclusions

There were two methods used in this investigation to measure throughput. First, a public domain network throughput measurement tool, New Test TCP *(nttcp)*, was used in order to minimize the workstation overhead. Next, the Remote Copy Protocol (*rcp)* system call was used in order to include all the overhead of daily distributed processing. The results obtained from these two test methods were consistent with each other.

New Test TCP (*nttcp*): During the *nttcp* tests the following tunable parameters were varied to determine their impact on throughput performance:

- TCP/IP window size, the amount of data that can be in transient at any one time between workstations.
- *sbf_num_llc_rx*, number of receive buffers (4k each) on the FDDI board allotted for receiving data.
- *nfs_async_threads*, number of asynchronous threads allotted for handling network file system service.
- *sbf_treq*, amount of time allotted for each workstation to transfer data prior to passing on the token. This is the TTRT.
- *sbf_mtu*, maximum protocol packet size.

Additionally, the *nttcp* tests were run on both single processor configurations and on two processor configurations. During this investigation the *nttcp* tests results showed that the four most significant impacts on throughput and the order of impact were as follows:

- Whether data was being transferred one-way or if both workstations were transferring data simultaneously.
- Whether the workstation had one or two processors
- The number of 4K receive buffers allotted for receiving data.
- The size of TCP/IP window available for sending data.

One note about the TCP/IP window size. During this investigation TCP/IP window sizes less than 20k and greater than 44k had too large of a deviation in their throughput results to be included in the final analysis. When the all of the TCP/IP window sizes are included, this parameter ends up having the largest impact on throughput rates. The rest of the results retain the above order of impact on throughput.

The other tunable parameters varied during these tests had little impact on throughput performance. Below are the rest of the factors affecting throughput in their order of importance:

- The length of the buffers being transmitted. This equates to the size of the data being transmitted.
- The Maximum Transmission Unit (MTU) size. This is the size of the FDDI frames of data being transmitted.
- The number of NFS asynchronous threads allowed for servicing network file service.
- The number of buffers (file size) being transmitted.

Remote Copy Protocol (*rcp*): During the *rcp* tests the tunable parameters were varied, but there was no noticeable difference in these throughput rates. The TCP/IP window size, which had the largest impact in the *nttcp* tests, did not have any noticable impact on throughput. The reason why the TCP/IP window did not have an impact was that *rcp* does small size read()'s and write()'s, so the all overhead dominates over the time spent in the kernel in TCP. If an application wants optimum bulk data throughput, it should increase the recieve buffering, and also do moderately large read()'s and write()'s so that the syscall overhead does not dominate.

The only difference between the *nttcp* tests and the *rcp* tests was the additional overhead with the *rcp* disk transfers and the *rcp* protocol overhead. Therefore, the conclusion can be drawn that one of these two differences accounted for the very large drop in throughput between the *nttcp* tests and the *rcp* tests.

On the transfer of a file size of over 17 Mbytes from White with two processors to Gold, the best achieved throughput rate was 13.54 Mbps with *rcp* when the transferred data is written to disk. This is only 13.54 percent of FDDI's available bandwidth and only 41.3 percent of the highest achieved throughput using *nttcp* at the same TCP/IP window size of 8k. Most of this 41.3 percent difference between *rcp* and *nttcp* can be attributed to the *rcp* protocol overhead. *RCP* has to go through a complete login, exec of the user's shell, and run through the user's ".cshrc" or ".profile" on the server side before it begins transfering any data. If the data transfer is not really huge, the time spent logging in will be much greater than the time spent transfering the data

B. TOPICS FOR FUTURE RESEARCH

Several topics for further study can be derived from this investigation. All of them are related to either improving throughput or to explaining events which were not explained in this thesis.

Since the *nttcp* tests were only able to obtain a maximum throughput using TCP transfers of 65 Mbps, 35 percent of the available bandwidth of FDDI is not being used. What portion of this unused bandwidth is due to lack of processor power and what portion is due to inefficiencies in the TCP/IP protocol?

This investigation primarily looked at throughput rates associated with TCP transfers, not User Datagram Protocol (UDP) transfers. The UDP frames have a header of 8 bytes and the TCP frames have a header of 20 bytes. Also, UDP is not a reliable transport protocol.

How much of a throughput can be achieved using UDP and what problems occur when using an unreliable transfer protocol?

File transfers using the rcp system call displayed a throughput rate of only 13.54 Mbps when the transferred data is written to disk. What percentage of this bottleneck is caused by the throughput rate on the SCSI-2 controller and what percentage is caused by other overhead associated with file transfers?

APPENDIX A: NTTCP PROGRAM and TEST SCRIPTS

DOIT.SH Script

```
#!/bin/sh

date > start
date > run1_start_time

ttest.sh 65536 512
ttest.sh 8192 512
ttest.sh 65536 1024
ttest.sh 8192 1024
ttest.sh 65536 2048
ttest.sh 8192 2048
ttest.sh 65536 4096
ttest.sh 8192 4096

date > run1_finish_time

mkdir run1
mv *.log *.out run1/.
mv *time run1/.

date > run2_start_time

ttest.sh 65536 512
ttest.sh 8192 512
ttest.sh 65536 1024
ttest.sh 8192 1024
ttest.sh 65536 2048
ttest.sh 8192 2048
ttest.sh 65536 4096
ttest.sh 8192 4096

date > run2_finish_time

mkdir run2
mv *.log *.out run2/.
mv *time run2/.

date > run3_start_time

ttest.sh 65536 512
ttest.sh 8192 512
ttest.sh 65536 1024
ttest.sh 8192 1024
ttest.sh 65536 2048
ttest.sh 8192 2048
ttest.sh 65536 4096
ttest.sh 8192 4096

date > run3_finish_time
mkdir run3
mv *.log *.out run3/.
mv *time run3/.

date > run4_start_time

ttest.sh 65536 512
ttest.sh 8192 512
ttest.sh 65536 1024
ttest.sh 8192 1024
ttest.sh 65536 2048
ttest.sh 8192 2048
ttest.sh 65536 4096
ttest.sh 8192 4096

date > run4_finish_time

mkdir run4
mv *.log *.out run4/.
mv *time run4/.

date > run5_start_time

ttest.sh 65536 512
ttest.sh 8192 512
ttest.sh 65536 1024
ttest.sh 8192 1024
ttest.sh 65536 2048
ttest.sh 8192 2048
ttest.sh 65536 4096
ttest.sh 8192 4096

date > run5_finish_time

mkdir run5
mv *.log *.out run5/.
mv *time run5/.
```

```
date > run6_start_time

ttest.sh 65536 512
ttest.sh 8192 512
ttest.sh 65536 1024
ttest.sh 8192 1024
ttest.sh 65536 2048
ttest.sh 8192 2048
ttest.sh 65536 4096
ttest.sh 8192 4096

date > run6_finish_time

mkdir run6
mv *.log *.out run6/.
mv *time run6/.

date > finish
```

TTEST.SH Script

```
#!/bin/sh
#
# Use nttcp to test network throughput.
# Usage: ttest.sh byte_per_write
                number_of_writes
#
DATALEN=$1
NPKTS=$2

#White to Gold
RECHOST=131.120.1.2
RSH=/usr/ucb/rsh
NTTCP=nttcp

rm -f ttest.out
rm -f ttest.tran.log
rm -f ttest.recv.log

# from 4KB to 60KB windows in steps of 8KB
SIZE=4
while test $SIZE -lt 61
do
      $RSH $RECHOST $NTTCP -r -w$SIZE
            > tmp1 2>&1 &
      sleep 5
      $NTTCP -t -l$DATALEN -n$NPKTS -w$SIZE
           $RECHOST >> ttest.tran.log 2>&1
      sleep 5
      grep 'Mb/s' tmp1 | awk '{print
'$SIZE'*1024,$12}' >> ttest.out
      cat tmp1 >> ttest.recv.log
     SIZE=`expr $SIZE + 8`
done

rm -f tmp1
mv ttest.out ttest.$DATALEN.$NPKTS.out
mv ttest.tran.log
     ttest.$DATALEN.$NPKTS.tran.log
mv ttest.recv.log
     ttest.$DATALEN.$NPKTS.recv.log
```

NTTCP Program

```
/*
 *      NTTCP.C
 *
 * Test TCP connection.  Makes a connection on port 2000
 * and transfers zero buffers or data copied from stdin.
 *
 * Usable on 4.2, 4.3, and 4.1a systems by defining one of
 * BSD42 BSD43 (BSD41a)
 *
 * Modified for operation under 4.2BSD, 18 Dec 84
 *      T.C. Slattery, USNA
 * Minor improvements, Mike Muuss and Terry Slattery, 16-Oct-85.
 *
 * Modified on 5 Apr 94 for opertion under Solaris 2.3 based on changes
 * for the TTCP.C program provided by Don Merritt of ARL.
 *      CPT Mark Schivley, USA
 */
#ifndef lint
static char RCSid[] = "@(#)$Header: /src/opt/brl/sbin/ttcp/RCS/ttcp.c,v 1.2 1993/11/30 20:15:39
root Exp $ (BRL)";
#endif
#define BSD43
/* #define BSD42 */
/* #define BSD41a */
#include <stdio.h>
#include <ctype.h>
#include <errno.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <sys/time.h>           /* struct timeval */
#ifdef SYSV
#include <sys/times.h>
#include <sys/param.h>
#else
#include <sys/resource.h>
#endif
#ifdef SYSV
#define bcopy(s,d,l) memcpy(d, s, (size_t) l)
#define bzero(s,l) memset(s, 0, (size_t) l)
#endif
struct sockaddr_in sinme;
struct sockaddr_in sinhim;
struct sockaddr_in sindum;
struct sockaddr_in frominet;
int domain, fromlen;
```

```
int fd;                        /* fd of network socket */
int sendwin = 32 * 1024;
int rcvwin = 32 * 1024;
int optlen = sizeof(int);
int buflen = 1024;             /* length of buffer */
char *buf;                     /* ptr to dynamic buffer */
int nbuf = 1024;               /* number of buffers to send in sinkmode */
int udp = 0;                   /* 0 = tcp, !0 = udp */
int options = 0;               /* socket options */
int one = 1;                   /* for 4.3 BSD style setsockopt() */
short port = 2001;             /* TCP port number */
char *host;                    /* ptr to name of host */
int trans;                     /* 0=receive, !0=transmit mode */
int sinkmode = 1;              /* 0=normal I/O, !0=sink/source mode */
int verbose = 0;
int nodelay = 0;               /* set TCP_NODELAY socket option */
int window = 0;                /* 0=use default   1=set to specified size*/
struct hostent *addr;
extern int errno;
char Usage[] = "\
Usage: ttcp -t [-options] host <in\n\
        -l##    length of bufs written to network (default 1024)\n\
        -s      don't source a pattern to network, use stdin\n\
        -n##    number of bufs written to network (-s only, default 1024)\n\
        -p##    port number to send to (default 2000)\n\
        -u      use UDP instead of TCP\n\
Usage: ttcp -r [-options] >out\n\
        -l##    length of network read buf (default 1024)\n\
        -s      sink (discard) all data from network\n\
        -p##    port number to listen at (default 2000)\n\
        -B      Only output full blocks, as specified in -l## (for TAR)\n\
        -u      use UDP instead of TCP\n\
";
char stats[128];
double t;                      /* transmission time */
long nbytes;                   /* bytes on net */
int b_flag = 0;                /* use mread() */
void prep_timer();
double read_timer();
double cput, realt;            /* user, real time (seconds) */
main(argc,argv)
int argc;
char **argv;
{
   unsigned long addr_tmp;
   if (argc < 2) goto usage;
   argv++; argc--;
   while( argc>0 && argv[0][0] == '-' ) {
   switch (argv[0][1]) {
   case 'B':
       b_flag = 1;
```

```
        break;
    case 't':
        trans = 1;
        break;
    case 'r':
        trans = 0;
        break;
    case 'd':
        options |= SO_DEBUG;
        break;
    case 'n':
        nbuf = atoi(&argv[0][2]);
        break;
    case 'l':
        buflen = atoi(&argv[0][2]);
        break;
    case 'w':
        window=1;
        sendwin = 1024 * atoi(&argv[0][2]);
        rcvwin = 1024 * atoi(&argv[0][2]);
        break;
    case 's':
        sinkmode = 1;/* source or sink, really */
        break;
    case 'p':
        port = atoi(&argv[0][2]);
        break;
    case 'u':
        udp = 1;
        break;
    default:
        goto usage;
    }
    argv++; argc--;
    }
    if(trans) {
    /* xmitr */
    if (argc != 1) goto usage;
    bzero((char *)&sinhim, sizeof(sinhim));
    host = argv[0];
    if (atoi(host) > 0 ) {
        /* Numeric */
        sinhim.sin_family = AF_INET;
#ifdef cray
        addr_tmp = inet_addr(host);
        sinhim.sin_addr = addr_tmp;
#else
        sinhim.sin_addr.s_addr = inet_addr(host);
#endif
    } else {
        if ((addr=gethostbyname(host)) == NULL)
```

```
            err("bad hostname");
        sinhim.sin_family = addr->h_addrtype;
        bcopy(addr->h_addr,(char*)&addr_tmp, addr->h_length);
#ifdef cray
        sinhim.sin_addr = addr_tmp;
#else
        sinhim.sin_addr.s_addr = addr_tmp;
#endif cray
    }
    sinhim.sin_port = htons(port);
    sinme.sin_port = 0;/* free choice */
    } else {
    /* rcvr */
    sinme.sin_port =  htons(port);
    }
    if( (buf = (char *)malloc(buflen)) == (char *)NULL)
    err("malloc");
    fprintf(stderr,"ttcp%s: nbuf=%d, buflen=%d, port=%d\n",
    trans?"-t":"-r",
    nbuf, buflen, port);
    if ((fd = socket(AF_INET, udp?SOCK_DGRAM:SOCK_STREAM, 0)) < 0)
    err("socket");
    mes("socket");
/* Try the getsockopt & setsockopt for Solaris here */
#ifndef SOLARIS
    if (bind(fd, &sinme, sizeof(sinme)) < 0)
    err("bind");
#else
    /*
     * Under Solaris, calling connect() on a stream socket binds the
     * socket to an address. If a bind() is done before the connect(),
     * an error "connect: Address family not supported by protocol family"
     * results. Only call bind() for the cases where you're not going
     * to call connect().
     */
    if (udp || (!udp && !trans) )
    if (bind(fd, (struct sockaddr *) &sinme, sizeof(sinme)) < 0)
        err("bind");
#endif /* SOLARIS */
    if (!udp)  {
      if (trans) {
    /* We are the client if transmitting */
    if(options)  {
#ifdef BSD42
        if( setsockopt(fd, SOL_SOCKET, options, 0, 0) < 0)
#else BSD43
#ifndef SOLARIS
        if( setsockopt(fd, SOL_SOCKET, options,  &one, sizeof(one)) < 0)
#else
        if( setsockopt(fd, SOL_SOCKET, options, (char *) &one, sizeof(one)) <
0)
```

```
#endif /* SOLARIS */
#endif
            err("setsockopt");
    }
#ifndef SOLARIS
    if(connect(fd, &sinhim, sizeof(sinhim) ) < 0) {
#else
    if(connect(fd, (struct sockaddr *) &sinhim, sizeof(sinhim) ) < 0) {
#endif /* SOLARIS */
          err("connect");
    }
    mes("connect");
   if(window){
    if (setsockopt (fd, SOL_SOCKET, SO_SNDBUF, (char *) &sendwin,
    sizeof(sendwin)) < 0 )
      printf("get send window size didn't work\n");
    if (setsockopt (fd, SOL_SOCKET, SO_RCVBUF, (char *) &rcvwin,
    sizeof(rcvwin)) < 0 )
     printf("get rcv window size didn't work\n");
    if (getsockopt (fd, SOL_SOCKET, SO_RCVBUF, (char *) &sendwin, &optlen) < 0 )
      printf("get send window size didn't work\n");
    else printf("send window size = %d\n", sendwin);
    if (getsockopt (fd, SOL_SOCKET, SO_RCVBUF, (char *) &rcvwin, &optlen) < 0 )
      printf("get rcv window size didn't work\n");
    else printf("receive window size = %d\n", rcvwin);
    }
      } else {
    /* otherwise, we are the server and
          * should listen for the connections
          */
#ifndef SOLARIS
    listen(fd,0);   /* allow a queue of 0 */
#else
      /*
       * Under Solaris, specifying a queue length of 0
       * results in a "connection refused".
       */
    listen(fd,1);
#endif /* SOLARIS */
    if(options)  {
#ifdef BSD42
          if( setsockopt(fd, SOL_SOCKET, options, 0, 0) < 0)
#else BSD43
#ifndef SOLARIS
          if( setsockopt(fd, SOL_SOCKET, options,  &one, sizeof(one)) < 0)
#else
          if( setsockopt(fd, SOL_SOCKET, options, (char *) &one, sizeof(one)) <
0)
#endif /* SOLARIS */
#endif
            err("setsockopt");
```

```
    }
    fromlen = sizeof(frominet);
    domain = AF_INET;
#ifndef SOLARIS
    if((fd=accept(fd, &frominet, &fromlen) ) < 0)
#else
    if((fd=accept(fd, (struct sockaddr *) &frominet, &fromlen) ) < 0)
#endif /* SOLARIS */
          err("accept");
    mes("accept");
    if (window){
    if (setsockopt (fd, SOL_SOCKET, SO_SNDBUF, (char *) &sendwin,
    sizeof(sendwin)) < 0 )
      printf("get send window size didn't work\n");
    if (setsockopt (fd, SOL_SOCKET, SO_RCVBUF, (char *) &rcvwin,
    sizeof(rcvwin)) < 0 )
     printf("get rcv window size didn't work\n");
    if (getsockopt (fd, SOL_SOCKET, SO_RCVBUF, (char *) &sendwin, &optlen) < 0 )
      printf("get send window size didn't work\n");
    else printf("send window size = %d\n", sendwin);
    if (getsockopt (fd, SOL_SOCKET, SO_RCVBUF, (char *) &rcvwin, &optlen) < 0 )
      printf("get rcv window size didn't work\n");
    else printf("receive window size = %d\n", rcvwin);
     }
     }
     }
    prep_timer();
    errno = 0;
    if (sinkmode) {
    register int cnt;
    if (trans) {
        pattern( buf, buflen );
        if(udp) (void)Nwrite( fd, buf, 4 ); /* rcvr start */
        while (nbuf-- && Nwrite(fd,buf,buflen) == buflen)
            nbytes += buflen;
        if(udp) (void)Nwrite( fd, buf, 4 ); /* rcvr end */
    } else {
        while ((cnt=Nread(fd,buf,buflen)) > 0) {
            static int going = 0;
            if( cnt <= 4 ) {
            if( going )
                break;/* "EOF" */
            going = 1;
            prep_timer();
            } else
            nbytes += cnt;
        }
    }
    } else {
    register int cnt;
    if (trans) {
```

```
        while((cnt=read(0,buf,buflen)) > 0 &&
          Nwrite(fd,buf,cnt) == cnt)
            nbytes += cnt;
    } else {
        while((cnt=Nread(fd,buf,buflen)) > 0 &&
          write(1,buf,cnt) == cnt)
            nbytes += cnt;
    }
    }
    if(errno) err("IO");
    (void)read_timer(stats,sizeof(stats));
    if(udp&&trans) {
    (void)Nwrite( fd, buf, 4 ); /* rcvr end */
    (void)Nwrite( fd, buf, 4 ); /* rcvr end */
    (void)Nwrite( fd, buf, 4 ); /* rcvr end */
    (void)Nwrite( fd, buf, 4 ); /* rcvr end */
    }
    fprintf(stdout,
      "ttcp%s: %ld bytes in %.2f real seconds = %.2f KB/sec = %.4f Mb/s\n",
         trans?"-t":"-r",
         nbytes, realt, ((double)nbytes)/realt/1024,
                  ((double)nbytes)/realt/128000 );
      if (verbose) {
        fprintf(stdout,
          "ttcp%s: %ld bytes in %.2f CPU seconds = %.2f KB/cpu sec\n",
          trans?"-t":"-r",
          nbytes, cput, ((double)nbytes)/cput/1024 );
    }
    exit(0);
usage:
    fprintf(stderr,Usage);
    exit(1);
}
err(s)
char *s;
{
    fprintf(stderr,"ttcp%s: ", trans?"-t":"-r");
    perror(s);
    fprintf(stderr,"errno=%d\n",errno);
    exit(1);
}
mes(s)
char *s;
{
    fprintf(stderr,"ttcp%s: %s\n", trans?"-t":"-r", s);
}
pattern( cp, cnt )
register char *cp;
register int cnt;
{
    register char c;
```

```
    c = 0;
    while( cnt-- > 0 ) {
    while( !isprint((c&0x7F)) ) c++;
    *cp++ = (c++&0x7F);
    }
}
/******* timing *********/
#ifdef SYSV
extern long time();
#if sgi
static void tvsub();
static structtimeval time0;/* Time at which timeing started */
#else
static long time0;
#endif
static struct tms tms0;
#else
static structtimeval time0;/* Time at which timeing started */
static structrusage ru0;/* Resource utilization at the start */
static void prusage();
static void tvadd();
static void tvsub();
static void psecs();
#endif
/*
 *     PREP_TIMER
 */
void
prep_timer()
{
#ifdef SYSV
#if sgi
   gettimeofday(&time0, (struct timezone *)0);
#else
   (void)time(&time0);
#endif
   (void)times(&tms0);
#else
   gettimeofday(&time0, (struct timezone *)0);
   getrusage(RUSAGE_SELF, &ru0);
#endif
}
/*
 *     READ_TIMER
 *
 */
double
read_timer(str,len)
char *str;
{
#ifdef SYSV
```

```
    long now;
    struct tms tmsnow;
    char line[132];
#ifdef sgi
    struct timeval timedol;
    struct timeval td;
    gettimeofday(&timedol, (struct timezone *)0);
    tvsub( &td, &timedol, &time0 );
    realt = td.tv_sec + ((double)td.tv_usec) / 1000000;
#else
    (void)time(&now);
    realt = now-time0;
#endif
    (void)times(&tmsnow);
    cput = tmsnow.tms_utime - tms0.tms_utime;
    cput /= HZ;
    if( cput < 0.00001 ) cput = 0.01;
    if( realt < 0.00001 ) realt = cput;
    sprintf(line,"%g CPU secs in %g elapsed secs (%g%%)",
    cput, realt,
    cput/realt*100 );
    (void)strncpy( str, line, len );
    return( cput );
#else
    /* BSD */
    struct timeval timedol;
    struct rusage ru1;
    struct timeval td;
    struct timeval tend, tstart;
    char line[132];
    getrusage(RUSAGE_SELF, &ru1);
    gettimeofday(&timedol, (struct timezone *)0);
    prusage(&ru0, &ru1, &timedol, &time0, line);
    (void)strncpy( str, line, len );
    /* Get real time */
    tvsub( &td, &timedol, &time0 );
    realt = td.tv_sec + ((double)td.tv_usec) / 1000000;
    /* Get CPU time (user+sys) */
    tvadd( &tend, &ru1.ru_utime, &ru1.ru_stime );
    tvadd( &tstart, &ru0.ru_utime, &ru0.ru_stime );
    tvsub( &td, &tend, &tstart );
    cput = td.tv_sec + ((double)td.tv_usec) / 1000000;
    if( cput < 0.00001 ) cput = 0.00001;
    return( cput );
#endif
}
#ifndef SYSV
static void
prusage(r0, r1, e, b, outp)
    register struct rusage *r0, *r1;
    struct timeval *e, *b;
```

```
    char *outp;
{
    struct timeval tdiff;
    register time_t t;
    register char *cp;
    register int i;
    int ms;
    t = (r1->ru_utime.tv_sec-r0->ru_utime.tv_sec)*100+
       (r1->ru_utime.tv_usec-r0->ru_utime.tv_usec)/10000+
       (r1->ru_stime.tv_sec-r0->ru_stime.tv_sec)*100+
       (r1->ru_stime.tv_usec-r0->ru_stime.ty_usec)/10000;
    ms = (e->tv_sec-b->tv_sec)*100 + (e->tv_usec-b->tv_usec)/10000;
#define END(x){while(*x) x++;}
    cp = "%Uuser %Ssys %Ereal %P %Xi+%Dd %Mmaxrss %F+%Rpf %Ccsw";
    for (; *cp; cp++) {
    if (*cp != '%')
        *outp++ = *cp;
    else if (cp[1]) switch(*++cp) {
    case 'U':
        tvsub(&tdiff, &r1->ru_utime, &r0->ru_utime);
        sprintf(outp,"%d.%01d", tdiff.tv_sec, tdiff.tv_usec/100000);
        END(outp);
        break;
    case 'S':
        tvsub(&tdiff, &r1->ru_stime, &r0->ru_stime);
        sprintf(outp,"%d.%01d", tdiff.tv_sec, tdiff.tv_usec/100000);
        END(outp);
        break;
    case 'E':
        psecs(ms / 100, outp);
        END(outp);
        break;
    case 'P':
        sprintf(outp,"%d%%", (int) (t*100 / ((ms ? ms : 1))));
        END(outp);
        break;
    case 'W':
        i = r1->ru_nswap - r0->ru_nswap;
        sprintf(outp,"%d", i);
        END(outp);
        break;
    case 'X':
        sprintf(outp,"%d", t == 0 ? 0 : (r1->ru_ixrss-r0->ru_ixrss)/t);
        END(outp);
        break;
    case 'D':
        sprintf(outp,"%d", t == 0 ? 0 :
           (r1->ru_idrss+r1->ru_isrss-(r0->ru_idrss+r0->ru_isrss))/t);
        END(outp);
        break;
    case 'K':
```

```
            sprintf(outp,"%d", t == 0 ? 0 :
              ((r1->ru_ixrss+r1->ru_isrss+r1->ru_idrss) -
              (r0->ru_ixrss+r0->ru_idrss+r0->ru_isrss))/t);
            END(outp);
            break;
        case 'M':
            sprintf(outp,"%d", r1->ru_maxrss/2);
            END(outp);
            break;
        case 'F':
            sprintf(outp,"%d", r1->ru_majflt-r0->ru_majflt);
            END(outp);
            break;
        case 'R':
            sprintf(outp,"%d", r1->ru_minflt-r0->ru_minflt);
            END(outp);
            break;
        case 'I':
            sprintf(outp,"%d", r1->ru_inblock-r0->ru_inblock);
            END(outp);
            break;
        case 'O':
            sprintf(outp,"%d", r1->ru_oublock-r0->ru_oublock);
            END(outp);
            break;
        case 'C':
            sprintf(outp,"%d+%d", r1->ru_nvcsw-r0->ru_nvcsw,
                r1->ru_nivcsw-r0->ru_nivcsw );
            END(outp);
            break;
        }
        }
        *outp = '\0';
}
static void
tvadd(tsum, t0, t1)
    struct timeval *tsum, *t0, *t1;
{
    tsum->tv_sec = t0->tv_sec + t1->tv_sec;
    tsum->tv_usec = t0->tv_usec + t1->tv_usec;
    if (tsum->tv_usec > 1000000)
     tsum->tv_sec++, tsum->tv_usec -= 1000000;
}
static void
tvsub(tdiff, t1, t0)
    struct timeval *tdiff, *t1, *t0;
{
    tdiff->tv_sec = t1->tv_sec - t0->tv_sec;
    tdiff->tv_usec = t1->tv_usec - t0->tv_usec;
    if (tdiff->tv_usec < 0)
     tdiff->tv_sec--, tdiff->tv_usec += 1000000;
```

```
}
static void
psecs(l,cp)
long l;
register char *cp;
{
    register int i;
    i = l / 3600;
    if (i) {
    sprintf(cp,"%d:", i);
    END(cp);
    i = l % 3600;
    sprintf(cp,"%d%d", (i/60) / 10, (i/60) % 10);
    END(cp);
    } else {
    i = l;
    sprintf(cp,"%d", i / 60);
    END(cp);
    }
    i %= 60;
    *cp++ = ':';
    sprintf(cp,"%d%d", i / 10, i % 10);
}
#endif
/*
 *      N R E A D
 */
Nread( fd, buf, count )
{
    struct sockaddr_in from;
    int len = sizeof(from);
    register int cnt;
    if( udp ) {
    cnt = recvfrom( fd, (char *) buf, count, 0, (struct sockaddr *) &from, &len );
    } else {
    if( b_flag )
         cnt = mread( fd, buf, count );/* fill buf */
    else
         cnt = read( fd, buf, count );
    }
    return(cnt);
}
/*
 *      N W R I T E
 */
Nwrite( fd, buf, count )
{
    register int cnt;
    if( udp ) {
again:
    cnt = sendto( fd, (char *) buf, count, 0, (struct sockaddr *) &sinhim,
```

```
sizeof(sinhim) );
    if( cnt<0 && errno == ENOBUFS ) {
        delay(18000);
        errno = 0;
        goto again;
    }
    } else {
    cnt = write( fd        ount );
    }
    return(cnt);
}
delay(us)
{
    struct timeval tv;
    tv.tv_sec = 0;
    tv.tv_usec = us;
    (void)select( 1, (fd_set *)0, (fd_set *)0, (fd_set *)0, &tv );
    return(1);
}
/*
 *      M R E A D
 *
 * This function performs the function of a read(II) but will
 * call read(II) multiple times in order to get the requested
 * number of characters. This can be necessary because
 * network connections don't deliver data with the same
 * grouping as it is written with. Written by Robert S. Miles, BRL.
 */
int
mread(fd, bufp, n)
int fd;
register char*bufp;
unsignedn;
{
    register unsignedcount = 0;
    register intnread;
    do {
    nread = read(fd, bufp, n-count);
    if(nread < 0)  {
        perror("ttcp_mread");
        return(-1);
    }
    if(nread == 0)
        return((int)count);
    count += (unsigned)nread;
    bufp += nread;
     } while(count < n);
    return((int)count);
}
#if sgi
static void
```

```
tvsub(tdiff, t1, t0)
    struct timeval *tdiff, *t1, *t0;
{
    tdiff->tv_sec = t1->tv_sec - t0->tv_sec;
    tdiff->tv_usec = t1->tv_usec - t0->tv_usec;
    if (tdiff->tv_usec < 0)
     tdiff->tv_sec--, tdiff->tv_usec += 1000000;
}
#endif
```

APPENDIX B: RCP PROGRAM

```
#include <stdio.h>
#include <sys/time.h>

main ()
{

 long elapsed_sec,     /* Seconds variable        */
     elapsed_usec;    /* Microseconds variable     */

 int  file_size;

 float total_time,
     part_usec,
     transfer_rate;

 float average_time = 0;

 int  loop_counter,
     a,           /* Subroutine result variables */
     b;
 int  n = 5;

 char name[30], system_name[30];
 char rcp_string[30] = "rcp";
 char blank_string[2] = " ";
 int  true = 1;
 char answer[2];
 char* get_name(char *string);


 /* Variable structure defns */

 struct timeval timestart, timedone;
 struct timezone zonestart, zonedone;

 /* Get file name & Dest machine name & path       */

 printf("\n\n\n Here is a list of availble files for transfering: \n\n");
 system ("ls -al");

 while(answer[0] != 'y')
  {
   printf("\n Input the file name to be transfered: \n\n");
   gets(name);
   printf("\n Is the below input correct? Enter y if yes or n if incorrect: \n\n");
```

```
   puts(name);
   printf("\n");
   gets(answer);
  }
answer[0] = 'n';     /* reset for next loop  */

/* Get file size                          */

while(answer[0] != 'y')
  {
   printf("\n Input the file size to be transfered: \n\n");
   scanf("%d", &file_size);
   printf("\n Is the below input correct? Enter y if yes or n if incorrect: \n\n");
   printf("%d\n", file_size);
   gets(answer);
   gets(answer);
answer[0] = 'y';
  }

answer[0] = 'n';   /* reset for next loop  */



while(answer[0] != 'y')
  {
   printf("\n Input the Dest machine name & path to be transfered: \n\n");
   printf("An example would be:  gold-fddi:/usr/test/wtog_test\n\n");
   gets(system_name);
   printf("\n Is the below input correct? Enter y if yes or n if incorrect: \n\n");
   puts(system_name);
   printf("\n");
   gets(answer);
  }

 strcat(rcp_string, blank_string);
 strcat(rcp_string, name);
 strcat(rcp_string, blank_string);
 strcat(rcp_string, system_name);

/*  Set up outer loop to execute transfers n times  */
for (loop_counter = 1; loop_counter <= n; loop_counter += 1)
  {
   /*  Get start time in sec&usec and check if successful */
  a = gettimeofday(&timestart, zonestart);
  if (a != 0)
    printf ("Oops ! %d\n", a);
   /*  Use system call to do file transfer  */
  system (rcp_string);
/*    system ("rcp american_pie.au gold-fddi:/usr/test/wtog_test"); */
   /*  Get stop time in sec&usec and check if successful */
  b = gettimeofday(&timedone, zonedone);
```

```
 if (b != 0)
   printf ("Oops! %d\n", b);
   /* Get structure values for calculations. */
 elapsed_sec = timedone.tv_sec - timestart.tv_sec;
 elapsed_usec = timedone.tv_usec - timestart.tv_usec;
   /* Make sure that we account for the usec */
   /* variable rooling over (through zero)  */
 if (elapsed_sec >= 1 )
 {
   if (elapsed_usec < 0)
   {
elapsed_sec -= 1;
elapsed_usec += 1000000;
   }
 }
   /* Convert the usec variable to a floating point number. */
 part_usec = elapsed_usec/1.0e6;
   /* Add the seconds to the microseconds to get a real number */
 total_time = elapsed_sec + part_usec;
   /* And print the results on the CRT */
 printf ("%f \t%f\n", total_time, ((file_size*8/total_time)/1000000));
 average_time =+ total_time;
}

/* Print out the results of the avg transfer rate */

printf("\n\nIs this time correct? %f", average_time);

   printf("\tThe average time was %f and the average transfer rate was %f\n", average_time/n,
((file_size*8/total_time)/1000000));

   /* This is the end of the control loop. */
 exit (0);
}
```

APPENDIX C: NEAL NELSON BENCHMARK RESULTS

TABLE 9: CPU SUBSYSTEM

GOLD2.SOL	White	Gold
CPU Type	Sparc	Sparc
CPU Clock Speed	45 MHz	50 MHz
Total Size of Main Memory	224 Mbytes	224 Mbytes
Speed of Main Memory Chips	80 ns	80 ns
Type and Speed of Math Coprocessor	None	None
Number of Main CPUs	2	2

TABLE 10: DISK SUBSYSTEM

	White	Gold
Total Number of Disk Controllers	1	1
Total Number of Disk Devices	2	2
Disk Drive Type	SCSI	SCSI
Disk Drive Brand/Model	Seagate	Seagate
Disk Average Seek Time Seagate ST11200 Seagate ST1480	 1-10.5ms 1-10.5ms	 2-10.5 ms
Does system have I/O buses separate from the main bus?	Yes	Yes

TABLE 11: CACHE INFORMATION

	White	Gold
Does the system have instruction or data cache?	Yes	Yes
How many levels of instruction/data cache are there?	2	2
How is cache coherency accomplished?	Snooping with invalidation	Snooping with invalidation
Does CPU have separate instruction and data caches?	Yes	Yes
Total size of all instructions/data caches: On-board Instruction Data (Note: External SuperCache controller provides 1 Mbyte external cache)	 20 Kbytes 16 Kbytes	 20 Kbytes 16 Kbytes
Total swap	approx 280 Mbytes	approx 280 Mbytes

Group 1: Tests a of mix of activities that are intended to approximate the processing activities for the following five types of users. Group 1 includes the following tests:

1) Simulated Office Automation Workload
2) Simulated Database Workload
3) Simulated Software Development Workload
4) Simulated Transaction Processing Workload
5) Simulated Calculation Workload (Math/Statistics/CAD/CAM)

Group 2: Tests designed to perform various types of calculation tasks and thereby profile the performance of the computer's calculation subsystem. Group 2 includes the following tests:

6) Write to Shared Memory

7) Read from Memory, Small Instruction Area, Small Data Area

8) Read from Memory, Small Instruction Area, Larger Data Area

9) Read from Memory, Larger Instruction Area, Small Data Area

10) Read from Memory, Larger Instruction Area, Larger Data Area

11) Make Machine Page or Swap with 'malloc' and 'free'

12) Combined Integer and Floating Point Math

13) Math Library Functions

14) Semaphores, Shared Memory, Context Switch

15) Write to and Read from Pipes, Context Switch

16) Sample System Calls

17) Increasing Depth of Function Calls

Group 3: Tests that perform a series of disk input and output functions to profile the performance of the disk subsystem. Group 3 includes the following tests:

18) 1024 byte Sequential Reads from Unix File(s)

19) 1024 byte Sequential Writes from Unix File(s)

20) 8192 byte Sequential Reads from Unix Files(s)

21) 8192 byte Sequential Writes to Unix File(s)

22) 4096 byte Synchronized Reads from Unix File(s)

23) 4096 byte Synchronized Reads from Raw Device(s)

24) 16384 byte Synchronized Reads from Unix File(s)

25) 16384 byte Synchronized Reads from Raw Device(s)

26) 4096 byte Pseudo Random Reads from Unix File(s)

27) 4096 byte Pseudo Random Reads from Raw Device(s)

28) Profile Disk Cache for Unix File(s)

29) Profile Disk Cache for Raw Device(s)

30) 8192 byte Sequential Writes then 'sync'

Gold Verses White, Two Processors

TABLE 12: GOLD2.SOL VRS WHITE2.SOL, TEST 1 & 2 & 3 & 4

	Test 1			Test 2			Test 3			Test 4	
Load	White Secs	Gold Secs		White Secs	Gold Secs		White Secs	Gold Secs		White Secs	Gold Secs
1	202	163		117	97		116	96		12	107
2	204	166		120	101		117	99		118	99
3	268	217		166	141		16	139		157	138
4	363	293		211	185		202	176		191	167
5	339	333		300	243		245	200		227	199
6	401	339		318	311		30	241		273	225
7	488	389		396	324		31	289		321	261
8	566	458		48	399		318	326		309	300
9	653	528		572	460		364	385		309	337
10	738	590		657	560		392	344		353	319
11	80	659		76	607		460	386		393	351
12	932	756		958	703		504	429		436	402
13	1012	841		1185	896		560	472		502	421
14	1132	928		1322	1023		606	516		542	508
15	1192	[illegible]		171	1286		647	556		588	496
16	1278	1068		180	1557		728	598		619	519
17	1378	1125		2004	1534		769	655		675	563
18	1540	1261		219	1789		822	666		694	599
19	1647	1351		2413	1920		902	727		805	655
20	1609	1404		2256	2112		92	750		800	650

TABLE 13: GOLD2.SOL VRS WHITE2.SOL, TEST 5 & 6 & 7 & 8

	Test 5			Test 6			Test 7			Test 8	
Load	White Secs	Gold Secs		White Secs	Gold Secs		White Secs	Gold Secs		White Secs	Gold Secs
1	203	164		128	103		130	106		130	105
2	206	167		128	104		131	106		133	107
3	273	221		192	155		195	160		205	166
4	333	298		254	204		257	211		281	243
5	367	319		318	260		322	259		399	335
6	477	398		378	307		391	311		514	436
7	585	477		432	353		447	363		641	539
8	701	570		487	400		507	408		765	641
9	798	641		540	443		551	452		846	722
10	931	737		612	493		611	500		994	831
11	999	818		676	547		678	554		1095	921
12	1157	926		739	611		748	620		1257	1069
13	1290	·1051		801	664		809	680		1359	1190
14	1377	1124		866	712		888	725		1522	1303
15	1707	1376		914	757		948	769		1905	1620
16	1953	1573		983	811		1005	815		2216	1844
17	2082	1693		1043	853		1067	881		2377	1996
18	2239	1786		1116	901		1124	926		2530	2147
19	2385	1912		1171	972		1198	982		2738	2279
20	2497	2025		1226	1024		1260	1040		2870	2458

TABLE 14: GOLD2.SOL VRS WHITE2.SOL, TEST 9 & 10 & 11 & 12

	Test 9			Test10			Test 11			Test 12	
Load	White Secs	Gold Secs		White Secs	Gold Secs		White Secs	Gold Secs		White Secs	Gold Secs
1	130	106		130	106		0	0		109	87
2	131	107		133	108		0	0		108	87
3	203	162		209	169		0	0		162	131
4	268	219		294	249		0	0		216	181
5	339	284		426	360		0	0		274	219
6	409	343		553	463		0	0		325	263
7	488	407		699	583		0	0		378	308
8	560	468		833	706		0	0		434	351
9	644	537		967	793		0	0		485	391
10	725	609		1106	939		0	0		540	436
11	807	697		1223	1044		0	0		596	488
12	887	773		1391	1200		0	0		649	523
13	986	834		1544	1318		0	0		718	569
14	1062	905		1726	1468		0	0		761	615
15	1141	983		2100	1847		0	0		810	653
16	1244	1052		2355	2023		0	0		866	704
17	1296	1130		2588	2139		0	0		921	745
18	1378	1190		2748	2330		0	0		986	787
19	1462	1278		2956	2472		0	0		1046	855
20	1543	1342		3067	2700		0	0		1105	885

TABLE 15: GOLD2.SOL VRS WHITE2.SOL, TEST 13 & 14 & 15 & 16

	Test 13			Test14			Test 15			Test 16	
Load	White Secs	Gold Secs		White Secs	Gold Secs		White Secs	Gold Secs		White Secs	Gold Secs
1	102	81		34	28		109	97		52	46
2	103	81		141	124		131	111		56	50
3	158	126		217	187		179	156		85	74
4	211	175		290	253		245	217		118	98
5	262	223		393	334		308	269		145	121
6	344	268		491	421		386	328		175	147
7	377	301		588	501		434	395		205	172
8	422	341		704	593		516	452		239	196
9	485	390		813	678		576	501		267	219
10	525	430		906	774		626	548		303	244
11	593	469		1013	868		686	611		329	280
12	685	535		1105	951		750	673		359	300
13	713	586		1205	1039		816	699		403	317
14	779	618		1296	1098		874	752		418	356
15	808	640		1410	1173		965	791		443	381
16	901	775		1541	1257		1008	839		471	383
17	897	747		1590	1384		1067	902		496	417
18	1008	820		1705	1416		1127	932		520	447
19	1061	863		1810	1513		1176	969		555	481
20	1118	932		1941	1602		1238	1118		612	490

TABLE 16: GOLD2.SOL VRS WHITE2.SOL, TEST 17 & 18 & 19 & 20

	Test 17			Test18			Test 19			Test 20	
Load	White Secs	Gold Secs		White Secs	Gold Secs		White Secs	Gold Secs		White Secs	Gold Secs
1	101	82		3	4		18	19		2	2
2	95	76	·	6	5		92	93		2	2
3	144	117		9	7		137	145		3	2
4	200	169		12	10		231	221		4	3
5	270	217		15	13		325	372		5	4
6	340	274		23	19		388	378		10	8
7	579	468		27	24		433	427		14	12
8	839	685		32	27		518	474		17	14
9	1077	869		37	30		540	539		19	17
10	1356	1083		39	34		655	574		21	18
11	1631	1314		43	38		691	662		24	20
12	1887	1533		49	42		694	727		28	22
13	2188	1761		332	334		1662	1230		300	320
14	2478	2005		298	273		1898	2316		229	426
15	2734	2213		254	442		2393	2149		204	252
16	3072	2631		266	260		1683	1714		257	246
17	3347	2707		274	278		1920	1865		268	270
18	3679	2939		288	295		2096	2046		282	284
19	3998	3364		309	299		3083	2247		300	299
20	4337	3574		324	324		2238	2139		326	331

TABLE 17: GOLD VRS WHITE2.SOL, TEST 21 & 22 & 23 & 24

	Test 21			Test 22			Test 23			Test 24	
Load	White Secs	Gold Secs	·	White Secs	Gold Secs		White Secs	Gold Secs		White Secs	Gold Secs
1	18	18		3	2		4	3		2	1
2	115	114		3	2		4	3		2	2
3	156	160		4	3		6	5		3	2
4	217	224		6	5		8	7		4	3
5	313	292		7	6		11	[illegible]		5	4
6	347	286		14	12		14	12		9	7
7	472	356		19	16		18	15		13	11
8	691	412		23	19		21	18		16	13
9	556	499		28	22		24	28		18	15
10	608	499		28	25		243	366		20	16
11	571	668		33	28		223	230		37	36
12	613	733		38	32		177	174		91	48
13	756	655		277	233		193	180		218	193
14	835	938		333	307		542	259		178	238
15	788	807		385	543		206	201		181	197
16	907	820		527	567		206	196		193	485
17	885	919		407	424		538	215		206	449
18	954	1037		483	478		241	236		500	370
19	951	1091		835	468		260	251		338	351
20	1171	1175		927	980		277	283		368	481

TABLE 18: GOLD2.SOL VRS WHITE2.SOL, TEST 25 & 26 & 27 & 28

	Test 25			Test 26			Test 27			Test 28	
Load	White Secs	Gold Secs		White Secs	Gold Secs		White Secs	Gold Secs		White Secs	Gold Secs
1	3	2		3	2		3	2		1	1
2	4	3		3	2		4	3		2	1
3	6	5		4	• 3		6	5		2	2
4	8	6		5	4		7	6		3	3
5	9	8		7	6		10	9		4	3
6	13	11		11	9		13	11		5	4
7	15	13		17	14		17	15		8	5
8	18	15		22	18		20	17		7	6
9	20	17		26	22		23	20		15	7
10	306	345		30	25		264	306		10	9
11	406	390		47	41		779	802		9	8
12	491	495		68	52		942	930		11	10
13	601	507		412	426		955	950		5	4
14	549	498		558	644		1163	1106		11	10
15	585	525		848	847		1251	1195		16	10
16	709	563		1128	1109		1354	1279		31	21
17	644	655		1259	1274		1549	1498		49	31
18	751	765		1570	1567		1608	1568		79	59
19	951	983		1701	1685		1788	1895		82	98
20	957	1006		1925	1897		1913	1916		116	106

TABLE 19: GOLD2.SOL VRS WHITE2.SOL, TEST 29 & 30

	Test 29			Test 30	
Load	White Secs	Gold Secs		White Secs	Gold Secs
1	1	1		1489	1405
2	1	1		1496	1382
3	1	1		154	1386
4	2	1		162	1511
5	2	2		1552	1583
6	3	3		1700	1713
7	4	3		2020	1745
8	4	5		2409	2069
9	4	4		2313	2498
10	2	4		2522	2766
11	38	39		3317	2781
12	62	59		274	2226
13	78	71		270	2249
14	89	97		2703	2268
15	100	101		2621	2533
16	106	107		2682	2733
17	120	131		2576	2694
18	139	125		3064	2661
19	142	133		3086	2666
20	146	156		2862	2786

Gold One Processor Verses Gold Two Processors Results

TABLE 20: GOLD1.SOL VRS GOLD2.SOL, TEST 1 & 2 & 3 & 4

	Test 1			Test 2			Test 3			Test 4	
Load	Gold1 Secs	Gold2 Secs		Gold1 Secs	Gold2 Secs		Gold1 Secs	Gold Secs		Gold1 Secs	Gold2 Secs
1	108	103		99	97		99	96		108	107
2	285	166		170	101		170	99		163	99
3	336	217		231	141		223	139		216	138
4	446	293		340	185		306	176		290	167
5	538	333		367	243		347	200		322	199
6	689	339		514	311		390	241		340	225
7	788	389		633	324		441	289		382	261
8	963	458		826	399		533	326		456	300
9	1050	528		917	460		600	385		493	337
10	1158	590		1095	560		680	344		549	319
11	1366	659		1260	607		767	386		629	351
12	1569	756		1472	703		903	429		735	402
13	1688	841		1650	896		981	472		773	421
14	1818	928		1839	1023		1063	516		824	508
15	1958	994		2379	1286		1153	556		911	496
16	2098	1068		2626	1557		1221	598		943	519
17	2230	1125		2835	1534		1298	655		1025	563
18	2383	1261		3201	1789		1360	666		1095	599
19	2512	1351		3368	1920		1484	727		1126	635
20	2672	1404		3600	2112		1553	750		1215	650

TABLE 21: GOLD1.SOL VRS GOLD2.SOL, TEST 5 & 6 & 7 & 8

	Test 5			Test 6			Test 7			Test 8	
Load	Gold1 Secs	Gold2 Secs		Gold1 Secs	Gold2 Secs		Gold1 Secs	Gold Secs		Gold1 Secs	Gold2 Secs
1	108	104		107	103		108	106		107	105
2	284	167		209	104		220	106		214	107
3	333	221		307	155		312	160		321	166
4	444	298		399	204		409	211		445	243
5	593	319		494	260		529	259		636	335
6	785	398		573	307		601	311		807	436
7	922	477		689	353		684	363		962	539
8	1155	570		753	400		769	408		1171	641
9	1251	641		851	443		868	452		1280	722
10	1456	737		941	493		965	500		1510	831
11	1675	818		1033	547		1052	554		1700	921
12	1967	926		1130	611		1130	620		1864	1069
13	2112	1051		1268	664		1321	680		2082	1190
14	2343	1124		1307	712		1333	725		2271	1303
15	2892	1376		1448	757		1450	769		2959	1620
16	3226	1573		1509	811		1539	815		3221	1844
17	3464	1693		1604	853		1679	881		3343	1996
18	3691	1786		1725	901		1744	926		3485	2147
19	3978	1912		1811	972		1832	982		3668	2279
20	4223	2025		1921	1024		1942	1040		3905	2458

TABLE 22: GOLD1.SOL VRS GOLD2.SOL, TEST 9 & 10 & 11 & 12

	Test 9			Test 10			Test 11			Test 12	
Load	Gold1 Secs	Gold2 Secs		Gold1 Secs	Gold2 Secs		Gold1 Secs	Gold Secs		Gold1 Secs	Gold2 Secs
1	112	106		107	106		0	0		88	87
2	219	107		221	108		0	0		177	87
3	323	162		337	169		0	0		265	131
4	435	219		481	249		0	0		353	181
5	545	284		679	360		0	0		445	219
6	663	343		881	463		0	0		523	263
7	783	407		1086	583		0	0		614	308
8	904	468		1296	706		0	0		689	351
9	1052	537		1459	793		0	0		801	391
10	1191	609		1725	939		0	0		894	436
11	1314	697		1922	1044		0	0		970	488
12	1392	773		2157	1208		0	0		1041	523
13	1583	834		2302	1318		0	0		1128	569
14	1643	905		2544	1468		0	0		1266	615
15	1750	983		2954	1847		0	0		1298	653
16	1848	1052		3267	2023		0	0		1404	704
17	1986	1130		3547	2139		0	0		1504	745
18	2088	1190		3797	2330		0	0		1623	787
19	2241	1278		4059	2472		0	0		1683	835
20	2338	1342		4312	2700		0	0		1767	885

TABLE 23: GOLD1.SOL VRS GOLD2.SOL, TEST 13 & 14 & 15 & 16

	Test 13			Test 14			Test 15			Test 16	
Load	Gold1 Secs	Gold2 Secs		Gold1 Secs	Gold2 Secs		Gold1 Secs	Gold Secs		Gold1 Secs	Gold2 Secs
1	85	81		28	28		88	97		45	46
2	212	81		101	124		172	111		92	50
3	298	126		167	187		263	156		137	74
4	405	175		277	253		344	217		181	98
5	483	223		357	334		442	269		230	121
6	589	268		447	421		531	328		272	147
7	643	301		523	501		609	395		319	172
8	717	341		608	593		703	452		362	196
9	856	390		710	678		776	501		416	219
10	932	430		782	774		869	548		466	244
11	1006	469		888	848		1016	611		510	280
12	1109	535		955	951		1072	673		549	300
13	1189	586		1056	1039		1135	699		598	317
14	1269	618		1128	1098		1207	752		662	356
15	1340	640		1227	1173		1283	791		703	381
16	1453	775		1376	1257		1364	839		799	383
17	1516	747		1408	1384		1438	902		814	417
18	1660	820		1489	1416		1504	932		850	447
19	1736	863		1581	1513		1649	969		914	481
20	1806	932		1659	1602		1739	1118		955	490

TABLE 24: GOLD1.SOL VRS GOLD2.SOL, TEST 17 & 18 & 19 & 20

	Test 17			Test 18			Test 19			Test 20	
Load	Gold1 Secs	Gold2 Secs		Gold1 Secs	Gold2 Secs		Gold1 Secs	Gold Secs		Gold1 Secs	Gold2 Secs
1	83	82		4	4		18	19		1	2
2	155	76		7	5		37	93		3	2
3	237	117		11	7		54	145		4	2
4	327	169		15	10		69	221		6	3
5	433	217		19	13		86	372		8	4
6	550	274		28	19		103	378		14	8
7	941	468		34	24		119	427		17	12
8	1345	685		37	27		135	474		19	14
9	1776	869		43	30		150	539		22	17
10	2386	1083		48	34		169	574		25	18
11	2814	1314		53	38		190	662		27	20
12	3024	1533		60	42		204	727		29	22
13	3561	1761		153	334		428	1230		47	320
14	3961	2085		198	273		437	2316		186	426
15	4359	2213		237	442		438	2149		148	252
16	5082	2631		248	260		470	1714		208	246
17	5465	2707		268	278		528	1865		238	270
18	6327	2939		280	295		544	2046		280	284
19	6379	3364		301	299		587	2247		291	299
20	6842	3574		321	324		674	2139		307	331

TABLE 25: GOLD1.SOL VRS GOLD2.SOL, TEST 21 & 22 & 23 & 24

	Test 21			Test 22			Test 23			Test 24	
Load	Gold1 Secs	Gold2 Secs		Gold1 Secs	Gold2 Secs		Gold1 Secs	Gold Secs		Gold1 Secs	Gold2 Secs
1	18	18		2	2		3	3		1	1
2	36	114		4	2		5	3		3	2
3	53	160		6	3		8	5		4	2
4	71	224		8	5		10	7		6	3
5	86	292		10	6		14	9		7	4
6	107	286		17	12		18	12		14	7
7	120	356		23	16		22	15		16	11
8	143	412		25	19		24	18		19	13
9	158	499		31	22		29	20		22	15
10	171	499		33	25		228	366		25	16
11	190	668		39	28		400	230		30	36
12	210	733		39	32		192	174		38	48
13	225	655		167	233		205	180		195	193
14	247	938		328	307		219	259		186	238
15	260	807		412	543		262	201		190	197
16	306	820		524	567		223	196		208	485
17	306	919		677	424		226	215		315	449
18	321	1037		646	478		242	236		279	378
19	346	1091		779	468		288	251		494	351
20	359	1175		1099	988		320	283		436	481

TABLE 26: GOLD1.SOL VRS GOLD2.SOL, TEST 25 & 26 & 27 & 28

	Test 25			Test 26			Test 27			Test 28	
Load	Gold1 Secs	Gold2 Secs		Gold1 Secs	Gold2 Secs		Gold1 Secs	Gold Secs		Gold1 Secs	Gold2 Secs
1	2	2		2	2		2	2		1	1
2	4	3		4	2		4	3		2	1
3	7	5		6	'3		7	5		3	2
4	9	6		8	4		10	6		4	3
5	12	8		10	6		13	9		6	3
6	15	11		15	9		16	'11		6	4
7	17	13		21	14		21	15		7	5
8	20	15		26	18		24	17		9	6
9	24	17		32	22		29	20		11	7
10	385	345		37	25		281	386		11	9
11	467	390		43	41		816	802		13	8
12	469	495		47	52		915	930		17	10
13	476	507		364	426		984	950		7	4
14	502	498		611	644		1105	1106		11	10
15	652	525		837	847		1201	1195		14	10
16	704	565		1035	1109		1315	1279		22	21
17	725	655		1313	1274		1469	1498		31	31
18	827	765		1485	1567		1680	1568		63	59
19	909	983		1678	1685		1783	1895		96	98
20	1012	1006		1798	1807		1959	1916		100	106

TABLE 27: GOLD1.SOL VRS GOLD2.SOL, TEST 29 & 30

	Test 29			Test 30	
Load	Gold1 Secs	Gold2 Secs		Gold1 Secs	Gold2 Secs
1	1	1		979	1485
2	1	1		965	1382
3	2	1		1088	1386
4	3	1		1114	1511
5	3	2		1197	1583
6	4	3		1296	1713
7	5	3		1564	1745
8	5	5		1688	2089
9	7	4		2067	2498
10	3	4		2321	2766
11	42	39		2419	2781
12	62	59		2190	2226
13	75	71		2229	2249
14	86	97		2284	2268
15	94	101		2525	2533
16	110	107		2693	2733
17	127	131		2657	2694
18	131	125		2674	2661
19	142	133		2668	2666
20	152	156		2746	2786

Solaris 2.3 One Processor Verses SunOS 4.1.3 One Processor Results

TABLE 28: GOLD1.SOL VRS GOLD1.SUN, TEST 1 & 2 & 3 & 4

	Test 1			Test 2			Test 3			Test 4	
Load	Gold Secs	Gold Secs		Gold Secs	Gold Secs		Gold Secs	Gold Secs		Gold Secs	Gold Secs
1	108	104		99	97		99	98		108	107
2	285	294		170	180		170	177		163	164
3	336	333		231	246		223	239		216	218
4	446	446		340	350		306	325		290	293
5	558	564		367	388		347	386		322	367
6	689	684		514	517		390	382		340	332
7	788	817		633	661		441	458		382	398
8	963	947		826	817		533	530		456	447
9	1050	1087		917	940		600	618		493	512
10	1158	1224		1095	1120		680	702		549	570
11	1366	1394		1260	1241		767	795		629	629
12	1569	1527		1472	1416		903	895		735	724
13	1688	1672		1650	1517		981	940		773	778
14	1818	1810		1859	1766		1063	1088		824	826
15	1958	1944		2379	2319		1153	1097		911	894
16	2088	2063		2626	2379		1221	1187		943	987
17	2230	2210		2835	2597		1298	1243		1025	1050
18	2383	2315		3201	2948		1360	1388		1095	1095
19	2512	2460		3368	3156		1484	1421		1126	1178
20	2672	2594		3600	3445		1553	1475		1215	1217

TABLE 29: GOLD1.SOL VRS GOLD1.SUN, TEST 5 & 6 & 7 & 8

	Test 5			Test 6			Test 7			Test 8	
Load	Gold Secs	Gold Secs		Gold Secs	Gold Secs		Gold Secs	Gold Secs		Gold Secs	Gold Secs
1	108	104		107	106		108	105		107	106
2	284	287		209	201		220	204		214	207
3	333	324		307	295		312	300		321	307
4	444	450		399	388		403	394		445	430
5	593	610		494	479		520	493		636	646
6	785	782		573	568		601	574		807	788
7	922	959		680	648		684	656		962	973
8	1135	1144		753	735		769	740		1171	1140
9	1251	1296		851	822		868	845		1280	1239
10	1456	1521		941	983		965	918		1510	1453
11	1675	1702		1053	980		1082	1013		1708	1637
12	1987	1801		1130	1064		1150	1100		1864	1773
13	2112	2072		1268	1157		1321	1185		2002	1981
14	2343	2291		1387	1247		1383	1272		2271	2243
15	2802	2942		1488	1376		1480	1403		2959	3057
16	3226	3179		1509	1431		1539	1489		3221	3180
17	3464	3371		1604	1531		1679	1583		3343	3362
18	3691	3614		1725	1633		1744	1629		3485	3501
19	3978	3923		1811	1719		1832	1700		3668	3752
20	4223	4182		1921	1780		1942	1825		3985	4063

TABLE 30: GOLD1.SOL VRS GOLD1.SUN, TEST 9 & 10 & 11 & 12

	Test 9			Test 10			Test 11			Test 12	
Load	Gold Secs	Gold Secs		Gold Secs	Gold Secs		Gold Secs	Gol Secs		Gold Secs	Gold Secs
1	112	106		107	108		0	0		88	87
2	219	219		221	224		0	0		177	174
3	323	326		337	337		0	0		265	38
4	435	435		481	466		0	0		353	346
5	545	548		679	670		0	0		445	431
6	663	654		881	855		0	0		523	517
7	783	767		1086	1070		0	0		614	601
8	904	884		1256	1267		0	0		689	686
9	1052	988		1459	1464		0	0		801	772
10	1191	1118		1725	1630		0	0		894	854
11	1314	1271		1922	1835		0	0		978	937
12	1392	1378		2157	2036		0	0		1041	1025
13	1583	1496		2302	2241		0	0		1128	1108
14	1643	1593		2544	2511		0	0		1266	1201
15	1750	1760		2954	3172		0	0		1298	1291
16	1848	1840		3267	3197		0	0		1404	1391
17	1986	1943		3547	3453		0	0		1504	1469
18	2088	2052		3797	3657		0	0		1623	1550
19	2241	2213		4059	3958		0	0		1683	1623
20	2338	2299		4312	4084		0	0		1767	1731

TABLE 31: GOLD1.SOL VRS GOLD1.SUN, TEST 13 & 14 & 15 & 16

	Test 13			Test 14			Test 15			Test 16	
Load	Gold Secs	Gold Secs		Gold Secs	Gold Secs		Gold Secs	Gold Secs		Gold Secs	Gold Secs
1	85	62		28	60		88	58		45	41
2	212	123		10	165		172	135		92	82
3	298	186		167	273		263	234		137	121
4	405	249		277	368		344	386		181	162
5	483	312		357	464		442	394		230	205
6	589	374		447	567		531	500		272	244
7	643	438		523	674		609	596		319	286
8	717	502		608	774		703	691		362	328
9	856	564		710	913		776	776		416	370
10	932	636		782	997		869	898		466	414
11	1006	749		888	1096		1016	995		510	454
12	1109	801		955	1237		1072	1083		549	497
13	1189	849		1056	1399		1133	1216		598	532
14	1269	956		1128	1429		1207	1260		662	597
15	1340	1040		1227	1585		1283	1486		703	629
16	1453	1101		1376	1723		1364	1415		799	667
17	1516	1129		1408	1895		1438	1506		814	718
18	1660	1230		1489	2010		1564	1612		850	765
19	1736	1283		1581	2088		1669	1719		914	816
20	1806	1399		1659	2132		1739	1815		955	838

TABLE 32: GOLD1.SOL VRS GOLD1.SUN, TEST 17 & 18 & 19 & 20

	Test 17			Test 18			Test 19			Test 20	
Load	Gold Secs	Gold Secs		Gold Secs	Gold Secs		Gold Secs	Gold Secs		Gold Secs	Gold Secs
1	83	78		4	4		18	14		1	2
2	155	149		7	8		37	29		3	5
3	237	224		11	11		54	44		4	7
4	327	311		15	15		69	59		6	10
5	433	409		19	19		86	73		8	12
6	550	525		28	23		103	89		14	15
7	941	1234		34	27		119	107		17	18
8	1345	1945		37	31		135	124		19	21
9	1776	2662		43	35		150	144		22	24
10	2386	3433		48	52		169	165		25	27
11	2814	4134		53	171		190	193		27	34
12	3024	4892		60	344		204	491		29	169
13	3561	5656		153	395		428	911		47	649
14	3961	6441		198	480		437	1064		186	769
15	4359	7058		237	656		438	1115		148	789
16	5082	7875		248	656		470	969		208	618
17	5465	8678		268	646		528	1283		238	1013
18	6327	9400		280	555		544	1530		280	1125
19	6379	10157		301	606		587	1835		291	1226
20	6842	11052		321	834		674	1895		307	1238

TABLE 33: GOLD1.SOL VRS GOLD1.SUN, TEST 21 & 22 & 23 & 24

	Test 21			Test 22			Test 23			Test 24	
Load	Gold Secs	Gold Secs		Gold Secs	Gold Secs		Gold Secs	Gold Secs		Gold Secs	Gold Secs
1	18	14		2	3		3	3		1	2
2	36	32		4	5		5	5		3	4
3	53	46		6	8		8	8		4	6
4	71	61		8	11		10	11		6	9
5	86	77		10	16		14	14		7	11
6	107	92		17	19		18	17		14	14
7	120	111		23	24		22	20		16	16
8	143	127		25	29		24	22		19	19
9	158	147		31	33		29	26		22	21
10	171	168		33	38		220	36		25	24
11	190	190		39	43		400	73		30	247
12	210	208		39	244		192	109		30	367
13	225	230		167	169		285	116		195	297
14	247	249		328	368		219	125		186	398
15	268	276		412	245		262	136		190	399
16	306	292		524	550		223	142		200	440
17	306	318		677	321		226	152		315	411
18	321	345		646	822		262	157		279	476
19	346	365		779	621		288	169		494	562
20	359	393		1099	1015		320	180		436	649

TABLE 34: GOLD1.SOL VRS GOLD1.SUN, TEST 25 & 26 & 27 & 28

	Test 25			Test 26			Test 27			Test 28	
Load	Gold Secs	Gold Secs		Gold1Secs	Gold Secs		Gold Secs	Gold Secs		Gold Secs	Gold Secs
1	2	2		2	3		2	2		1	1
2	4	4		4	4		4	4		2	1
3	7	5		6	7		7	7		3	3
4	9	8		8	11		10	10		4	3
5	12	10		10	17		13	14		6	4
6	15	12		15	21		16	17		6	5
7	17	14		21	26		21	19		7	6
8	20	16		26	30		24	23		9	7
9	24	18		32	35		29	26		11	8
10	305	21		37	40		281	29		11	10
11	467	22		43	709		816	32		13	43
12	469	237		47	1174		915	39		17	131
13	476	285		364	1336		984	74		7	193
14	502	224		611	1687		1105	19		11	221
15	652	130		837	1837		1201	56		14	259
16	704	152		1035	1934		1315	1122		22	277
17	725	225		1313	1964		1469	317		31	296
18	827	193		1455	2086		1680	368		63	319
19	909	168		1678	2361		1783	2248		96	349
20	1012	184		1798	2449		1959	2453		100	389

TABLE 35: GOLD1.SOL VRS GOLD1.SUN, TEST 29 & 30

	Test 29			Test 30	
Load	Gold Secs	Gold Secs		Gold Secs	Gold Secs
1	1	1		979	789
2	1	2		965	474
3	2	3		1008	625
4	3	4		1114	736
5	3	5		1197	785
6	4	8		1296	858
7	5	8		1564	1087
8	5	11		1688	1196
9	7	33		2067	1357
10	3	388		2321	1368
11	42	11		2419	1367
12	62	7		2190	1361
13	75	10		2229	1268
14	86	5		2284	1311
15	94	10		2525	1344
16	110	6		2693	1331
17	127	102		2657	1382
18	131	112		2674	1464
19	142	195		2668	1468
20	152	202		2746	1559

APPENDIX D: NTTCP SINGLE PROCESSOR RESULTS

TABLE 36: SINGLE PARAMETER TEST RESULTS

Test Number	From:	To:	NFS_asynch_threads	TTRT	sbr_num_llc_rx
1st Test	White	Gold	8	8ms	48K
2nd Test	Gold	White	8	8ms	48K
3rd Test & 4thTest	White Gold	Gold White	8	8ms	48K
5th Test	White	Gold	16	8ms	48K
6th Test	Gold	White	16	8ms	48K
7th Test & 8th Test	White Gold	Gold White	16	8ms	48K
9th Test	White	Gold	8	5ms	48K
10th Test	Gold	White	8	5ms	48K
11th Test 12th Test	White Gold	Gold White	8	5ms	48K
13th Test	White	Gold	16	5ms	48K
14th Test	Gold	White	16	5ms	48K
15th Test 16th Test	White Gold	Gold White	16	5ms	48K
17th Test	White	Gold	8	11ms	48K
18th Test	Gold	White	8	11ms	48K
19th Test & 20th Test	White Gold	Gold White	8	11ms	48K
21st Test	White	Gold	16	11ms	48K
22nd Test	Gold	White	16	11ms	48K
23rd Test & 24th Test	White Gold	Gold White	16	11ms	48K
25th Test	White	Gold	8	25ms	48K
26st Test	Gold	White	8	25ms	48K
27th Test & 28th Test	White Gold	Gold White	8	25ms	48K
29th Test	White	Gold	16	25ms	48K
30th Test	Gold	White	16	25ms	48K
31st Test & 32nd Test	White Gold	Gold White	16	25ms	48K
33rd Test FDDI Boards switched	White	Gold	8	8	48K
34th Test FDDI Boards switched	Gold	White	8	8	48K

TABLE 37: SINGLE PROCESSOR, 1ST TEST RESULTS

Window Size (K bytes)	File A Mbps	File B Mbps	File C Mbps	File D Mbps	File E Mbps	File F Mbps	File G Mbps	File H Mbps
4	32.77	38.23	40.05	36.06	32.46	31.92	31.51	31.96
12	118.33	29.13	32.77	30.95	24.63	25.42	24.93	26.21
20	32.77	43.69	41.87	40.57	40.57	40.33	40.62	39.86
28	32.77	49.15	38.23	42.65	40.57	40.89	41.67	41.81
36	32.77	43.69	38.23	43.69	41.61	40.89	41.67	42.38
44	32.77	49.15	38.23	42.65	40.57	39.43	42.26	42.09
52	76.96	49.15	38.23	41.61	38.75	37.93	39.35	36.15
60	32.77	43.69	38.23	41.61	33.72	34.37	30.09	30.60

From: White　Threads: 8　LLC Buffers: 48K
To: Gold　TTRT: 8ms　Single Test

TABLE 38: SINGLE PROCESSOR, 2ND TEST RESULTS

Window Size (K bytes)	File A Mbps	File B Mbps	File C Mbps	File D Mbps	File E Mbps	File F Mbps	File G Mbps	File H Mbps
4	32.77	32.77	32.77	34.33	30.34	29.47	30.12	29.98
12	30.04	29.13	29.49	30.34	23.08	21.41	21.16	24.09
20	32.77	30.95	38.23	33.55	33.55	34.59	33.33	34.57
28	32.77	43.69	34.59	36.67	35.89	35.84	35.56	35.85
36	32.77	38.23	36.41	36.67	34.02	34.64	35.37	35.25
44	32.77	49.15	36.41	37.45	35.11	35.00	34.80	35.37
52	163.84	32.77	40.05	35.28	14.42	14.58	13.45	14.21
60	32.77	43.69	32.77	36.67	7.79	7.04	7.56	6.93

From: Gold　Threads: 8　LLC Buffers: 48K
To: White　TTRT: 8ms　Single Test

TABLE 39: SINGLE PROCESSOR, 3RD TEST RESULTS

Window Size (K bytes)	File A Mbps	File B Mbps	File C Mbps	File D Mbps	File E Mbps	File F Mbps	File G Mbps	File H Mbps
4	30.04	25.49	29.34	26.55	19.58	20.98	18.93	20.79
12	27.31	26.40	23.35	25.73	18.06	20.37	17.58	19.44
20	24.58	27.31	26.99	29.61	28.29	28.94	28.43	27.37
28	102.60	33.68	26.42	25.57	24.81	28.75	29.03	29.47
36	30.04	32.77	28.61	28.09	29.65	29.49	29.92	27.91
44	30.04	41.87	28.76	30.41	25.92	30.08	30.08	29.29
52	30.04	34.59	32.25	30.97	24.08	25.94	28.24	25.58
60	32.77	28.22	31.68	30.03	26.60	26.37	24.03	25.23

From: White　Threads: 8　LLC Buffers: 48K
To: Gold　TTRT: 8ms　Dual Test

TABLE 40: SINGLE PROCESSOR, 4TH TEST RESULTS

Window Size (K bytes)	File A Mbps	File B Mbps	File C Mbps	File D Mbps	File E Mbps	File F Mbps	File G Mbps	File H Mbps
4	67.36	27.31	30.58	28.26	20.59	20.35	19.34	20.68
12	27.31	29.13	25.67	25.75	19.40	18.77	18.81	18.98
20	24.58	30.95	26.94	27.87	27.53	26.52	25.20	24.40
28	32.77	29.13	28.76	27.81	25.92	26.26	27.00	25.97
36	32.77	34.59	30.58	28.78	28.47	27.76	26.69	26.17
44	30.04	38.23	31.68	31.38	27.32	27.69	28.15	28.74
52	27.31	23.67	32.40	30.99	14.88	15.30	14.03	14.75
60	27.31	32.77	26.42	32.51	8.58	8.44	7.95	41.07

From: Gold　　Threads: 8　　LLC Buffers: 48K
To: White　　TTRT: 8ms　　Dual Test

TABLE 41: SINGLE PROCESSOR, 5TH TEST RESULTS

Window Size (K bytes)	File A Mbps	File B Mbps	File C Mbps	File D Mbps	File E Mbps	File F Mbps	File G Mbps	File H Mbps
4	32.77	32.77	38.23	36.67	33.55	32.49	32.44	32.95
12	30.04	32.77	34.59	34.40	26.51	26.17	27.10	27.53
20	32.77	43.69	38.23	40.57	39.53	39.85	40.14	41.26
28	128.04	43.69	41.87	40.57	41.61	42.01	42.28	41.95
36	32.77	38.23	41.87	41.61	41.61	42.09	42.53	41.69
44	136.53	38.23	43.69	42.65	40.57	42.57	42.53	41.98
52	32.77	43.69	38.23	42.65	38.49	38.11	39.00	35.48
60	32.77	49.15	40.05	40.57	32.94	35.84	34.70	33.80

From: White　　Threads: 16　　LLC Buffers: 48K
To: Gold　　TTRT: 8ms　　Single Test

TABLE 42: SINGLE PROCESSOR, 6TH TEST RESULTS

Window Size (K bytes)	File A Mbps	File B Mbps	File C Mbps	File D Mbps	File E Mbps	File F Mbps	File G Mbps	File H Mbps
4	32.77	43.69	31.68	33.55	29.13	29.47	30.13	29.03
12	32.77	30.95	27.31	29.13	21.28	23.01	24.97	22.51
20	95.57	32.77	31.68	33.72	33.72	33.54	34.95	33.52
28	81.92	38.23	34.59	35.89	35.89	35.78	35.57	34.62
36	32.77	49.15	34.59	37.45	33.55	35.42	35.96	35.40
44	105.33	38.23	34.59	36.67	33.72	34.74	34.06	34.41
52	95.57	43.69	34.59	35.89	12.57	12.03	10.84	10.90
60	30.04	36.41	34.59	34.33	6.86	6.52	6.24	6.17

From: Gold　　Threads: 16　　LLC Buffers: 48K
To: White　　TTRT: 8ms　　Single Test

TABLE 43: SINGLE PROCESSOR, 7TH TEST RESULTS

Window Size (K bytes)	File A Mbps	File B Mbps	File C Mbps	File D Mbps	File E Mbps	File F Mbps	File G Mbps	File H Mbps
4	32.77	23.67	28.40	23.96	21.84	21.81	21.76	19.39
12	24.58	30.95	24.08	23.60	17.60	18.03	20.25	18.84
20	87.38	30.04	26.03	22.41	26.15	26.58	28.29	30.81
28	30.04	32.77	26.60	24.93	31.45	25.68	29.00	30.81
36	32.77	32.77	36.04	29.30	27.02	27.77	29.50	29.26
44	30.04	36.41	28.24	29.13	28.17	29.14	30.14	29.11
52	30.04	30.95	32.40	29.09	24.96	25.41	30.13	28.46
60	74.22	30.95	29.73	28.37	28.24	25.39	26.77	25.60

From: White　　Threads: 16　　LLC Buffers: 48K
To: Gold　　TTRT: 8ms　　Dual Test

TABLE 44: SINGLE PROCESSOR, 8TH TEST RESULTS

Window Size (K bytes)	File A Mbps	File B Mbps	File C Mbps	File D Mbps	File E Mbps	File F Mbps	File G Mbps	File H Mbps
4	27.31	27.31	29.86	29.72	21.90	19.40	18.04	20.86
12	32.77	25.49	26.94	27.15	18.86	17.60	15.65	16.14
20	30.04	25.49	30.58	28.88	24.93	26.16	24.07	27.03
28	30.04	29.13	30.06	30.65	28.90	25.74	25.32	28.52
36	32.77	38.23	30.95	30.64	23.87	26.27	26.92	29.95
44	63.59	27.31	25.75	34.40	23.72	26.29	27.56	29.50
52	92.84	24.58	32.77	29.44	11.60	11.49	12.76	12.36
60	30.04	32.77	28.52	30.44	6.81	7.29	7.12	7.14

From: Gold　　Threads: 16　　LLC Buffers: 48K
To: White　　TTRT: 8ms　　Dual Test

TABLE 45: SINGLE PROCESSOR, 9TH TEST RESULTS

Window Size (K bytes)	File A Mbps	File B Mbps	File C Mbps	File D Mbps	File E Mbps	File F Mbps	File G Mbps	File H Mbps
4	32.77	32.77	40.05	35.28	34.33	34.28	33.32	33.84
12	32.77	36.41	31.68	32.33	25.18	25.74	26.67	25.12
36	70.09	43.69	38.23	40.83	38.49	39.99	40.10	40.14
44	163.84	43.69	38.23	40.57	41.61	41.53	41.99	41.84
52	105.33	43.69	43.69	42.65	39.53	42.57	42.26	42.09
60	63.72	38.23	43.69	42.65	40.83	42.01	41.67	41.85
60	32.77	38.23	43.69	43.69	37.71	35.84	37.67	35.03
60	32.77	38.23	40.05	41.87	28.08	31.95	30.55	28.37

From: White　　Threads: 8　　LLC Buffers: 48K
To: Gold　　TTRT: 5ms　　Single Test

TABLE 46: SINGLE PROCESSOR, 10TH TEST RESULTS

Window Size (K bytes)	File A Mbps	File B Mbps	File C Mbps	File D Mbps	File E Mbps	File F Mbps	File G Mbps	File H Mbps
4	32.77	38.23	37.14	33.72	29.13	28.72	28.90	28.24
12	32.77	30.95	30.58	31.24	25.99	26.49	24.57	25.87
20	32.77	38.23	34.59	35.11	34.50	33.54	33.84	30.69
28	81.92	32.77	36.41	33.72	35.89	32.14	32.43	28.56
36	63.72	38.23	38.23	35.11	31.45	31.36	32.77	27.85
44	32.77	38.23	38.23	35.89	32.23	30.41	32.58	28.65
52	32.77	38.23	36.41	35.89	18.37	16.68	17.16	14.76
60	95.57	32.77	34.59	34.33	10.29	9.33	10.37	10.62

From: Gold Threads: 8 LLC Buffers: 48K
To: White TTRT: 5ms Single Test

TABLE 47: SINGLE PROCESSOR, 11TH TEST RESULTS

Window Size (K bytes)	File A Mbps	File B Mbps	File C Mbps	File D Mbps	File E Mbps	File F Mbps	File G Mbps	File H Mbps
4	24.58	25.85	27.51	28.37	17.43	22.19	18.28	21.07
12	30.04	26.40	23.69	24.29	17.16	18.86	17.80	18.53
20	24.58	32.77	28.76	31.14	23.90	27.62	26.91	28.62
28	27.31	40.05	26.99	30.69	28.87	28.22	30.12	27.73
36	27.31	30.40	27.67	31.78	27.12	28.88	30.55	27.66
44	30.04	28.22	28.24	32.41	27.48	28.94	29.32	27.23
52	26.40	36.41	31.68	33.41	24.90	26.21	26.51	26.90
60	32.77	25.49	26.94	31.92	25.05	24.15	23.11	26.35

From: White Threads: 8 LLC Buffers: 48K
To: Gold TTRT: 5ms Dual Test

TABLE 48: SINGLE PROCESSOR, 12TH TEST RESULTS

Window Size (K bytes)	File A Mbps	File B Mbps	File C Mbps	File D Mbps	File E Mbps	File F Mbps	File G Mbps	File H Mbps
4	30.04	30.95	29.49	25.93	21.57	20.89	18.25	20.29
12	32.77	25.49	24.45	24.10	21.28	18.79	16.74	19.28
20	23.67	30.95	25.69	29.36	26.48	24.24	24.88	26.26
28	30.04	43.69	24.60	29.68	27.36	26.13	25.99	27.23
36	66.32	24.58	27.12	30.41	31.24	25.13	25.63	27.29
44	90.11	30.95	29.86	30.50	27.36	27.06	27.42	28.61
52	30.04	30.95	27.31	32.94	14.03	13.82	14.52	13.27
60	30.04	27.31	29.13	29.68	7.64	7.39	7.99	7.37

From: Gold Threads: 8 LLC Buffers: 48K
To: White TTRT: 5ms Dual Test

TABLE 49: SINGLE PROCESSOR, 13TH TEST RESULTS

Window Size (K bytes)	File A Mbps	File B Mbps	File C Mbps	File D Mbps	File E Mbps	File F Mbps	File G Mbps	File H Mbps
4	32.77	32.77	36.41	35.11	32.94	33.13	32.95	33.14
12	32.77	30.95	36.41	33.24	25.91	27.67	27.67	26.42
20	81.92	49.15	43.69	38.49	39.53	40.33	41.14	40.86
28	105.33	38.23	40.05	42.65	40.57	41.05	42.55	42.66
36	32.77	43.69	43.69	43.69	40.57	42.57	42.26	42.67
44	185.69	49.15	40.05	43.69	40.57	43.13	40.95	42.66
52	105.33	49.15	43.69	44.11	38.49	37.98	39.83	39.73
60	132.18	54.61	41.87	42.65	36.67	34.18	37.50	32.25

From: White　　Threads: 16　　LLC Buffers: 48K
To: Gold　　TTRT: 5ms　　Single Test

TABLE 50: SINGLE PROCESSOR, 14TH TEST RESULTS

Window Size (K bytes)	File A Mbps	File B Mbps	File C M	File D Mbps	File E Mbps	File F Mbps	File G Mbps	File H Mbps
4	32.77	38.23	3[illegible]	35.89	30.34	29.86	29.55	30.18
12	27.31	32.77	29.49	29.25	25.18	25.72	24.92	24.77
20	191.75	38.23	31.68	35.89	33.72	33.54	34.78	34.86
28	32.77	38.23	34.59	36.67	35.28	34.95	35.39	35.56
36	87.99	43.69	31.68	36.67	33.11	34.95	35.76	36.07
44	32.77	43.69	34.59	35.11	32.16	35.37	35.56	35.29
52	81.92	32.77	32.77	37.45	15.45	15.[illegible]3	15.98	14.90
60	32.77	32.77	32.77	35.89	8.64	7.45	7.87	7.25

From: Gold　　Threads: 16　　LLC Buffers: 48K
To: White　　TTRT: 5ms　　Single Test

TABLE 51: SINGLE PROCESSOR, 15TH TEST RESULTS

Window Size (K bytes)	File A Mbps	File B Mbps	File C Mbps	File D Mbps	File E Mbps	File F Mbps	File G Mbps	File H Mbps
4	30.04	29.13	25.49	24.90	19.39	18.90	20.87	20.88
12	24.58	21.85	24.03	21.71	15.54	15.74	17.77	17.35
20	30.04	27.31	27.51	27.81	25.49	25.35	25.31	25.58
28	30.04	36.41	31.68	23.05	26.24	24.51	27.38	25.99
36	32.77	34.59	28.76	29.54	30.15	25.61	27.72	27.23
44	32.77	41.87	31.68	34.66	31.54	26.63	27.74	27.85
52	30.04	32.77	32.40	29.93	29.98	24.58	25.73	28.30
60	30.04	34.59	29.86	30.20	28.26	22.75	22.52	24.28

From: White　　Threads: 16　　LLC Buffers: 48K
To: Gold　　TTRT: 5ms　　Dual Test

TABLE 52: SINGLE PROCESSOR, 16TH TEST RESULTS

Window Size (K bytes)	File A Mbps	File B Mbps	File C Mbps	File D Mbps	File E Mbps	File F Mbps	File G Mbps	File H Mbps
4	27.31	33.68	25.21	28.53	21.70	19.56	21.78	19.03
12	30.04	28.22	26.42	22.17	22.02	20.35	19.08	19.66
20	30.04	30.95	29.34	25.57	26.82	23.18	23.95	24.41
28	32.77	29.13	24.42	29.13	26.60	24.88	26.10	25.81
36	30.04	36.41	28.76	27.81	25.20	24.02	26.76	25.70
44	24.58	30.95	28.40	27.81	27.76	25.20	26.26	25.67
52	32.77	34.59	26.60	28.87	13.07	14.29	13.63	13.95
60	32.77	29.13	33.37	30.95	7.36	8.01	8.58	8.33

From: Gold   Threads: 16   LLC Buffers: 48K
To: White   TTRT: 5ms   Dual Test

TABLE 53: SINGLE PROCESSOR, 17TH TEST RESULTS

Window Size (K bytes)	File A Mbps	File B Mbps	File C Mbps	File D Mbps	File E Mbps	File F Mbps	File G Mbps	File H Mbps
4	30.04	36.41	36.41	35.89	31.68	33.17	33.32	33.05
12	32.77	32.77	36.41	36.67	27.36	29.96	27.08	27.11
20	168.13	43.69	40.05	38.75	39.53	39.37	39.49	39.99
28	32.77	43.69	40.05	41.61	39.53	43.13	41.00	42.53
36	32.77	32.77	40.05	42.65	41.87	40.41	41.56	42.23
44	163.84	38.23	41.87	39.79	40.83	42.57	43.11	41.78
52	32.77	54.61	41.87	39.79	35.28	34.29	36.22	38.24
60	253.31	43.69	40.05	40.83	29.91	31.85	34.62	30.08

From: White   Threads: 8   LLC Buffers: 48K
To: Gold   TTRT: 11ms   Single Test

TABLE 54: SINGLE PROCESSOR, 18TH TEST RESULTS

Window Size (K bytes)	File A Mbps	File B Mbps	File C Mbps	File D Mbps	File E Mbps	File F Mbps	File G Mbps	File H Mbps
4	32.77	32.77	36.41	36.67	31.55	30.01	30.12	29.84
12	32.77	30.95	28.40	27.76	25.45	24.21	23.51	24.12
20	61.44	38.23	34.59	36.67	35.11	34.32	34.40	34.49
28	118.33	32.77	38.23	37.10	35.11	36.25	35.96	35.97
36	32.77	32.77	38.23	35.89	35.89	36.62	35.58	35.76
44	32.77	38.23	36.41	36.67	35.89	35.42	36.20	35.19
52	69.32	32.77	38.23	37.45	16.28	15.68	18.83	14.83
60	32.77	54.61	36.41	35.28	8.92	8.11	8.33	7.13

From: Gold   Threads: 8   LLC Buffers: 48K
To: White   TTRT: 11ms   Single Test

TABLE 55: SINGLE PROCESSOR, 19TH TEST RESULTS

Window Size (K bytes)	File A Mbps	File B Mbps	File C Mbps	File D Mbps	File E Mbps	File F Mbps	File G Mbps	File H Mbps
4	30.04	24.58	21.05	24.90	21.80	21.92	21.04	21.36
12	24.58	30.95	22.26	23.96	18.24	19.59	21.64	18.60
20	32.77	32.77	25.69	26.51	27.29	28.42	27.54	29.22
28	30.04	27.31	26.58	28.24	33.41	26.04	29.59	29.77
36	32.77	27.67	28.76	27.75	33.11	26.92	30.67	29.45
44	79.19	36.41	27.67	28.85	30.15	27.21	28.95	28.52
52	30.04	36.41	36.41	27.05	28.76	26.75	27.56	26.48
60	30.04	34.59	29.86	25.61	30.06	25.97	24.51	23.74

From: White Threads: 8 LLC Buffers: 48K
To: Gold TTRT: 11ms Dual Test

TABLE 56: SINGLE PROCESSOR, 20TH TEST RESULTS

Window Size (K bytes)	File A Mbps	File B Mbps	File C Mbps	File D Mbps	File E Mbps	File F Mbps	File G Mbps	File H Mbps
4	133.80	30.95	31.31	26.86	23.60	19.96	20.18	21.23
12	32.77	28.22	25.85	26.19	22.68	18.91	16.92	18.46
20	32.77	29.13	28.76	29.72	27.97	22.66	24.06	25.52
28	30.04	27.31	28.24	28.14	28.87	24.44	25.14	28.26
36	90.11	35.50	28.40	29.72	30.36	26.22	24.20	28.82
44	32.77	38.23	29.49	31.14	28.74	24.76	26.79	28.91
52	84.35	27.31	27.51	31.02	14.08	13.99	12.68	13.08
60	29.13	30.95	32.40	29.72	8.80	8.24	7.74	7.60

From: Gold Threads: 8 LLC Buffers: 48K
To: White TTRT: 11ms Dual Test

TABLE 57: SINGLE PROCESSOR, 21ST TEST RESULTS

Window Size (K bytes)	File A Mbps	File B Mbps	File C Mbps	File D Mbps	File E Mbps	File F Mbps	File G Mbps	File H Mbps
4	32.77	32.77	38.23	34.33	32.16	31.48	31.97	33.17
12	30.04	32.77	32.77	34.50	24.63	27.89	27.05	25.93
20	32.77	43.69	36.41	39.53	36.93	39.43	39.09	39.86
28	136.53	43.69	38.23	40.57	39.01	40.63	39.72	42.11
36	136.53	43.69	43.69	41.61	40.83	39.51	39.00	41.95
44	136.53	38.23	41.87	40.57	39.01	40.55	38.91	41.28
52	32.77	38.23	41.87	41.61	36.58	38.77	33.83	40.30
60	32.77	49.15	38.23	43.69	32.16	31.82	29.50	30.95

From: White Threads: 16 LLC Buffers: 48K
To: Gold TTRT: 11ms Single Test

TABLE 58: SINGLE PROCESSOR, 22ND TEST RESULTS

Window Size (K bytes)	File A Mbps	File B Mbps	File C Mbps	File D Mbps	File E Mbps	File F Mbps	File G Mbps	File H Mbps
4	32.77	32.77	34.59	36.67	29.13	29.16	29.54	29.35
12	30.04	32.77	32.77	34.33	24.44	24.92	23.94	22.26
20	63.72	38.23	34.59	35.11	33.55	33.86	34.39	33.23
28	29.13	43.69	38.23	37.45	34.33	34.64	34.99	35.17
36	32.77	43.69	38.23	35.89	32.77	34.64	34.61	35.85
44	72.82	32.77	36.41	35.89	33.55	35.37	34.99	35.75
52	32.77	49.15	38.23	35.89	14.02	14.08	14.24	14.37
60	30.04	32.77	38.23	35.89	8.09	7.82	7.03	6.93

From: Gold Threads: 16 LLC Buffers: 48K
To: White TTRT: 11ms Single Test

TABLE 59: SINGLE PROCESSOR, 23RD TEST RESULTS

Window Size (K bytes)	File A Mbps	File B Mbps	File C Mbps	File D Mbps	File E Mbps	File F Mbps	File G Mbps	File H Mbps
4	30.04	30.04	29.49	25.57	21.70	19.28	20.61	20.01
12	27.31	27.31	24.24	24.63	18.06	18.14	16.77	16.71
20	27.31	29.13	26.42	23.72	26.53	25.37	24.93	27.40
28	30.04	29.13	29.34	25.57	31.01	28.52	27.70	29.68
36	26.40	38.23	28.76	27.15	30.77	27.12	28.27	27.80
44	32.77	34.59	32.40	30.36	34.66	29.42	29.66	29.73
52	27.31	30.95	31.68	33.67	28.64	27.95	27.30	26.68
60	133.80	34.59	31.68	32.02	25.33	23.90	24.70	24.02

From: White Threads: 16 LLC Buffers: 48K
To: Gold TTRT: 11ms Dual Test

TABLE 60: SINGLE PROCESSOR, 24TH TEST RESULTS

Window Size (K bytes)	File A Mbps	File B Mbps	File C Mbps	File D Mbps	File E Mbps	File F Mbps	File G Mbps	File H Mbps
4	133.80	40.96	27.67	26.72	22.18	20.34	21.30	20.30
12	30.04	28.22	24.76	25.05	20.82	22.10	20.67	20.47
20	30.04	32.77	29.34	29.27	25.49	28.41	25.45	26.34
28	81.92	29.13	32.77	30.06	25.78	27.02	26.38	26.00
36	32.77	27.31	28.97	28.05	26.55	26.62	25.74	26.12
44	30.04	31.86	34.59	29.96	25.99	27.71	26.01	27.89
52	27.31	29.13	27.15	27.81	15.88	16.08	14.63	16.43
60	30.04	41.87	27.91	29.15	9.39	9.50	8.59	8.95

From: Gold Threads: 16 LLC Buffers: 48K
To: White TTRT: 11ms Dual Test

TABLE 61: SINGLE PROCESSOR, 25TH TEST RESULTS

Window Size (K bytes)	File A Mbps	File B Mbps	File C Mbps	File D Mbps	File E Mbps	File F Mbps	File G Mbps	File H Mbps
4	87.99	32.77	34.59	35.11	31.55	30.59	31.63	31.83
12	32.77	38.23	32.77	33.72	28.28	30.88	24.70	26.55
20	32.77	43.69	40.05	42.65	41.61	40.41	41.14	40.33
28	32.77	32.77	40.05	41.87	43.69	42.01	42.55	42.38
36	105.33	43.69	41.87	42.65	43.69	42.57	41.67	42.09
44	81.92	43.69	41.87	41.61	41.61	41.45	42.23	42.23
52	32.77	49.15	43.69	40.83	38.75	37.58	39.35	38.44
60	32.77	49.15	40.05	41.61	35.89	35.11	34.75	30.13

From: White　　Threads: 8　　LLC Buffers: 48K
To: Gold　　TTRT: 25ms　　Single Test

TABLE 62: SINGLE PROCESSOR, 26TH TEST RESULTS

Window Size (K bytes)	File A Mbps	File B Mbps	File C Mbps	File D Mbps	File E Mbps	File F Mbps	File G Mbps	File H Mbps
4	32.77	38.23	32.77	34.33	29.13	29.16	29.43	29.34
12	32.77	30.95	31.68	31.55	26.55	26.01	27.20	25.66
20	79.19	32.77	32.77	36.67	34.33	34.22	34.58	33.94
28	72.82	36.41	36.41	37.45	35.89	35.78	35.57	34.96
36	32.77	32.77	32.77	35.28	34.33	35.42	35.96	35.46
44	32.77	32.77	34.59	36.67	34.33	35.37	34.87	35.06
52	32.77	32.77	34.59	37.45	19.49	17.71	17.49	16.59
60	32.77	32.77	35.32	34.33	8.42	8.54	8.36	8.31

From: Gold　　Threads: 8　　LLC Buffers: 48K
To: White　　TTRT: 25ms　　Single Test

TABLE 63: SINGLE PROCESSOR, 27T[illegible] ST RESULTS

Window Size (K bytes)	File A Mbps	File B Mbps	File C Mbps	File D Mbps	File E Mbps	File F Mbps	File G Mbps	File H Mbps
4	27.31	26.40	25.33	23.65	20.05	19.52	20.50	17.46
12	30.04	21.30	23.30	23.39	21.25	17.41	15.50	15.28
20	27.31	32.77	31.16	25.99	30.22	24.59	22.45	26.98
28	27.31	28.22	30.43	25.94	32.23	26.52	27.08	29.58
36	30.04	47.33	28.76	27.15	32.94	28.14	27.63	29.92
44	32.77	28.22	30.95	33.11	31.63	30.47	30.05	30.40
52	30.04	36.41	35.32	32.80	27.39	30.66	27.91	27.93
60	102.60	38.23	29.49	31.75	25.66	29.02	25.58	23.69

From: White　　Threads: 8　　LLC Buffers: 48K
To: Gold　　TTRT: 25ms　　Dual Test

TABLE 64: SINGLE PROCESSOR, 28TH TEST RESULTS

Window Size (K bytes)	File A Mbps	File B Mbps	File C Mbps	File D Mbps	File E Mbps	File F Mbps	File G Mbps	File H Mbps
4	78.28	27.31	27.12	26.99	17.56	20.76	21.53	20.65
12	32.77	32.77	24.27	25.21	21.19	21.80	19.80	20.10
20	32.77	27.67	28.61	28.76	24.46	24.38	24.76	26.12
28	32.77	34.59	24.21	27.15	28.14	26.36	26.74	26.47
36	30.04	29.13	29.49	25.17	26.96	25.85	26.93	25.70
44	32.77	30.95	25.85	29.99	25.99	26.06	27.33	26.33
52	67.36	24.58	27.18	28.74	15.88	15.51	14.99	14.10
60	95.57	30.95	28.24	29.56	8.73	8.66	8.49	8.91

From: Gold Threads: 8 LLC Buffers: 48K
To: White TTRT: 25ms Dual Test

TABLE 65: SINGLE PROCESSOR, 29TH TEST RESULTS

Window Size (K bytes)	File A Mbps	File B Mbps	File C Mbps	File D Mbps	File E Mbps	File F Mbps	File G Mbps	File H Mbps
4	32.77	38.23	34.59	35.89	31.68	32.49	32.79	32.14
12	32.77	32.77	34.59	32.33	27.48	29.08	28.00	26.95
20	32.77	43.69	40.05	40.57	39.53	41.45	40.60	40.86
28	105.33	43.69	43.69	40.57	40.57	42.01	41.67	41.67
36	81.92	43.69	41.87	43.69	40.83	40.41	41.94	41.83
44	135.62	49.15	43.69	42.65	41.61	41.53	42.53	42.24
52	30.04	49.15	43.69	42.65	32.94	38.59	37.68	36.14
60	81.92	49.15	41.87	43.69	35.11	35.59	33.86	30.88

From: White Threads: 16 LLC Buffers: 48K
To: Gold TTRT: 25ms Single Test

TABLE 66: SINGLE PROCESSOR, 30TH TEST RESULTS

Window Size (K bytes)	File A Mbps	File B Mbps	File C Mbps	File D Mbps	File E Mbps	File F Mbps	File G Mbps	File H Mbps
4	32.77	32.77	34.59	35.89	29.13	29.78	29.84	30.32
12	32.77	32.77	29.49	32.16	27.27	26.78	26.37	24.84
20	32.77	38.23	34.59	35.11	35.11	33.90	34.76	34.59
28	32.77	38.23	34.59	37.45	36.67	36.20	35.96	35.41
36	32.77	43.69	34.59	35.11	35.89	36.20	35.58	35.46
44	32.77	49.15	34.59	35.89	35.89	35.78	35.97	35.96
52	69.32	43.69	38.23	36.67	16.94	15.80	15.77	16.73
60	32.77	30.95	34.59	36.67	7.85	8.08	8.53	7.31

From: Gold Threads: 16 LLC Buffers: 48K
To: White TTRT: 25ms Single Test

TABLE 67: SINGLE PROCESSOR, 31ST TEST RESULTS

Window Size (K bytes)	File A Mbps	File B Mbps	File C Mbps	File D Mbps	File E Mbps	File F Mbps	File G Mbps	File H Mbps
4	24.58	29.13	27.31	27.12	25.21	21.11	18.25	20.21
12	110.14	27.31	25.69	24.44	20.41	19.38	19.74	19.99
20	30.04	22.76	33.50	29.68	28.33	29.28	28.41	27.24
28	24.58	33.13	23.87	27.35	29.82	31.37	30.80	29.15
36	32.77	29.13	30.95	33.70	29.63	32.85	29.96	28.01
44	30.04	30.95	28.40	32.91	32.63	30.73	29.57	28.26
52	32.77	38.23	30.38	32.49	26.44	26.24	26.29	27.34
60	30.04	25.49	28.42	29.06	26.27	25.55	23.73	25.90

From: White | Threads: 16 | LLC Buffers: 48K
To: Gold | TTRT: 25ms | Dual Test

TABLE 68: SINGLE PROCESSOR, 32ND TEST RESULTS

Window Size (K bytes)	File A Mbps	File B Mbps	File C Mbps	File D Mbps	File E Mbps	File F Mbps	File G Mbps	File H Mbps
4	30.04	31.86	28.03	26.75	20.72	19.48	19.31	19.47
12	27.31	29.13	26.21	24.13	20.00	16.33	17.72	18.47
20	23.67	30.95	27.70	29.02	27.03	25.25	25.38	24.82
28	30.04	28.22	31.16	30.23	28.42	26.06	25.82	25.73
36	23.67	29.13	30.06	28.50	26.68	26.97	25.30	27.19
44	27.31	30.95	31.68	28.80	27.06	27.23	26.22	27.90
52	87.99	27.31	25.33	29.14	15.18	14.30	14.79	14.26
60	21.85	29.13	27.67	30.13	7.79	8.34	8.33	7.93

From: Gold | Threads: 16 | LLC Buffers: 48K
To: White | TTRT: 25ms | Dual Test

TABLE 69: SINGLE PROCESSOR, 33RD TEST RESULTS

Window Size (K bytes)	File A Mbps	File B Mbps	File C Mbps	File D Mbps	File E Mbps	File F Mbps	File G Mbps	File H Mbps
4	32.77	32.77	38.23	35.11	32.33	32.49	32.29	33.48
12	32.77	43.69	40.05	34.50	27.12	29.79	31.03	27.60
20	137.14	54.61	43.69	39.53	39.53	39.37	41.16	40.22
28	69.32	38.95	41.37	39.79	41.61	40.97	42.82	41.54
36	32.77	32.77	41.37	42.65	40.83	40.89	40.93	39.95
44	87.99	43.69	43.69	43.69	31.05	33.05	35.01	35.42
52	32.77	49.15	41.37	39.01	15.12	12.40	14.07	13.06
60	69.32	36.41	40.05	41.61	13.01	11.07	12.96	13.06

From: White | Threads: 8 | LLC Buffers: 48K
To: Gold | TTRT: 8ms | Single Test, FDDI Boards Switched

TABLE 70: SINGLE PROCESSOR, 34TH TEST RESULTS

Window Size (K bytes)	File A Mbps	File B Mbps	File C Mbps	File D Mbps	File E Mbps	File F Mbps	File G Mbps	File H Mbps
4	32.77	38.23	36.41	32.77	28.16	28.64	28.86	29.13
12	32.77	38.23	30.58	31.12	20.13	20.72	19.77	20.56
20	32.77	32.77	32.77	32.77	32.77	34.22	33.48	33.05
28	72.82	38.23	34.59	35.11	31.35	33.13	32.33	33.35
36	32.77	32.77	36.41	35.89	22.50	25.98	23.71	23.74
44	32.77	38.95	31.68	32.46	6.76	5.51	6.35	5.66
52	17.29	20.39	21.87	22.77	4.11	4.17	4.22	3.91
60	11.38	12.56	14.59	17.10	3.54	3.48	3.48	3.53

From: Gold   Threads: 8   LLC Buffers: 48K

To: White   TTRT: 8ms   Single Test, FDDI Boards Switched

APPENDIX E: NTTCP TWO PROCESSORS RESULTS

TABLE 71: PARAMETERS USED FOR TWO PROCESSOR TEST

Test Number	From:	To:	NFS_asynch_threads	TTRT	sbf_num_llc_rx	sbf_mtu
1st Test	White	Gold	8	8ms	48K	4352
2nd Test	White	Gold	16	8ms	48K	4352
3rd Test	White	Gold	8	5ms	48K	4352
4th Test	White	Gold	16	5ms	48K	4352
5th Test	White	Gold	8	11ms	48K	4352
6th Test	White	Gold	16	11ms	48K	4352
7th Test	White	Gold	8	25ms	48K	4352
8th Test	White	Gold	16	25ms	48K	4352
9th Test	White	Gold	8	8ms	56K	4352
10th Test	White	Gold	16	8ms	56K	4352
11th Test	White	Gold	8	5ms	56K	4352
12th Test	White	Gold	16	5ms	56K	4352
13th Test	White	Gold	8	11ms	56K	4352
14th Test	White	Gold	16	11ms	56K	4352
15th Test	White	Gold	8	25ms	56K	4352
16th Test	White	Gold	16	25ms	56K	4352
17th Test	White	Gold	8	8ms	40K	4352
18th Test	White	Gold	16	8ms	40K	4352
19th Test	White	Gold	8	5ms	40K	4352
20th Test	White	Gold	16	5ms	40K	4352
21st Test	White	[illegible]	8	11ms	40K	4352
22nd Test	White	[illegible]	16	11ms	40K	4352
23th Test	White	Gold	8	25ms	40K	4352
24th Test	White	Gold	16	25ms	40K	4352
25th Test	White	Gold	8	8ms	48K	4192
26th Test	White	Gold	16	8ms	48K	4192
27th Test	White	Gold	8	5ms	48K	4192
28th Test	White	Gold	16	5ms	48K	4192
29th Test	White	Gold	8	11ms	48K	4192

TABLE 71: PARAMETERS USED FOR TWO PROCESSOR TEST

Test Number	From:	To:	NFS_asynch_threads	TTRT	sbf_num_llc_rx	sbf_mtu
30th Test	White	Gold	16	11ms	48K	4192
31st Test	White	Gold	8	25ms	48K	4192
32nd Test	White	Gold	16	25ms	48K	4192
33rd Test	White	Gold	8	8ms	56K	4192
34th Test	White	Gold	16	8ms	56K	4192
35th Test	White	Gold	8	5ms	56K	4192
36th Test	White	Gold	16	5ms	56K	4192
37th Test	White	Gold	8	11ms	56K	4192
38th Test	White	Gold	16	11ms	56K	4192
39th Test	White	Gold	8	25ms	56K	4192
40th Test	White	Gold	16	25ms	56K	4192
41st Test	White	Gold	8	8ms	40K	4192
42nd Test	White	Gold	16	8ms	40K	4192
43rd Test	White	Gold	8	5ms	40K	4192
44th Test	White	Gold	16	5ms	40K	4192
45th Test	White	Gold	8	11ms	40K	4192
46th Test	White	Gold	16	11ms	40K	4192
47th Test	White	Gold	8	25ms	40K	4192
48th Test	White	Gold	16	25ms	40K	4192

TABLE 72: TWO PROCESSORS, 1ST TEST RESULTS

Window Size (K bytes)	File A Mbps	File B Mbps	File C Mbps	File D Mbps	File E Mbps	File F Mbps	File G Mbps	File H Mbps
4	30.04	32.77	32.77	32.77	30.34	31.80	30.71	31.08
12	30.04	32.77	31.31	29.37	30.46	30.34	30.43	30.86
20	353.17	60.07	54.61	54.61	50.97	52.43	50.84	51.36
28	746.38	54.61	58.25	58.98	49.52	52.43	52.47	52.88
36	105.33	60.07	54.61	55.34	50.97	50.84	52.47	53.12
44	32.77	49.15	58.25	52.43	49.52	51.63	52.52	53.33
52	32.77	43.69	50.97	56.80	48.06	51.63	52.06	52.28
60	502.44	54.61	50.97	53.16	50.97	46.71	50.80	47.17

From: White Threads: 8 LLC Buffers: 48K
To: Gold TTRT: 8ms MTU: 4352 Bytes

TABLE 73: TWO PROCESSORS, 2ND TEST RESULTS

Window Size (K bytes)	File A Mbps	File B Mbps	File C Mbps	File D Mbps	File E Mbps	File F Mbps	File G Mbps	File H Mbps
4	32.77	32.77	32.77	33.55	31.55	31.16	31.77	31.54
12	27.31	32.77	29.49	30.95	31.55	29.98	30.27	30.69
20	32.77	54.61	50.97	50.97	49.52	51.63	50.76	52.22
28	32.77	60.07	54.61	53.16	50.97	51.63	52.47	51.68
36	266.70	60.07	47.33	55.34	50.97	52.61	52.89	52.88
44	367.47	54.61	47.33	54.61	50.97	54.55	52.52	53.60
52	240.30	60.07	50.97	58.98	48.06	53.40	50.76	52.25
60	32.77	49.15	43.69	50.97	49.52	49.25	52.10	44.91

From: White Threads: 16 LLC Buffers: 48K
To: Gold TTRT: 8ms MTU: 4352 Bytes

TABLE 74: TWO PROCESSORS, 3RD TEST RESULTS

Window Size (K bytes)	File A Mbps	File B Mbps	File C Mbps	File D Mbps	File E Mbps	File F Mbps	File G Mbps	File H Mbps
4	32.77	32.77	32.77	35.11	30.95	32.13	31.46	31.70
12	30.04	30.95	30.58	30.95	30.34	30.55	31.02	30.55
20	118.33	60.07	58.25	54.61	52.43	50.05	49.59	50.97
28	573.44	43.69	58.25	52.43	52.43	51.63	51.60	51.21
36	32.77	43.69	58.25	52.43	50.97	54.37	52.89	53.11
44	303.10	60.07	58.25	56.80	49.52	51.63	52.52	52.67
52	385.93	60.07	54.61	53.16	47.02	50.05	49.59	50.70
60	95.57	54.61	50.97	56.80	49.20	47.79	44.75	44.41

From: White Threads: 8 LLC Buffers: 48K
To: Gold TTRT: 5ms MTU: 4352 Bytes

TABLE 75: TWO PROCESSORS, 4TH TEST RESULTS

Window Size (K bytes)	File A Mbps	File B Mbps	File C Mbps	File D Mbps	File E Mbps	File F Mbps	File G Mbps	File H Mbps
4	32.77	32.77	32.77	32.77	32.16	31.48	31.95	32.18
12	32.77	32.77	30.58	31.68	31.24	31.20	34.76	31.95
20	458.49	54.61	50.97	56.80	50.97	52.61	52.09	52.25
28	95.57	54.61	54.61	54.61	53.16	52.43	53.35	51.89
36	136.53	65.54	54.61	56.80	50.97	52.61	55.70	54.48
44	32.77	60.07	58.25	53.16	52.43	51.63	53.39	53.56
52	249.40	43.69	58.25	56.80	48.06	50.84	50.42	52.23
60	281.91	49.15	54.61	58.98	48.06	50.84	47.54	46.71

From: White Threads: 16 LLC Buffers: 48K
To: Gold TTRT: 5ms MTU: 4352 Bytes

TABLE 76: TWO PROCESSORS, 5TH TEST RESULTS

Window Size (K bytes)	File A Mbps	File B Mbps	File C Mbps	File D Mbps	File E Mbps	File F Mbps	File G Mbps	File H Mbps
4	30.04	32.77	32.77	34.33	30.95	31.16	31.95	31.94
12	27.31	27.31	30.58	30.95	33.01	29.16	30.45	30.18
20	230.94	60.07	58.25	53.16	46.60	50.18	51.60	50.35
28	32.77	60.07	54.61	52.43	52.43	50.18	51.64	51.80
36	32.77	54.61	54.61	54.61	53.16	48.82	52.47	52.68
44	398.68	41.87	54.61	61.17	52.43	52.43	52.52	52.08
52	136.53	38.23	54.61	54.61	51.70	51.61	50.08	51.45
60	32.77	54.61	58.25	54.61	52.12	47.54	46.37	45.30

From: White Threads: 8 LLC Buffers: 48K
To: Gold TTRT: 11ms MTU: 4352 Bytes

TABLE 77: TWO PROCESSORS, 6TH TEST RESULTS

Window Size (K bytes)	File A Mbps	File B Mbps	File C Mbps	File D Mbps	File E Mbps	File F Mbps	File G Mbps	File H Mbps
4	32.77	30.95	34.59	33.55	32.16	32.45	32.11	32.10
12	27.31	30.95	31.68	32.16	30.95	32.13	30.86	31.23
20	300.37	54.61	54.61	56.80	52.43	51.63	50.80	51.25
28	163.84	49.15	58.25	56.80	54.61	52.43	50.80	52.92
36	190.89	49.15	54.61	58.98	53.16	51.02	49.46	52.75
44	435.74	65.54	58.25	56.80	52.43	50.84	53.01	54.06
52	476.96	60.07	54.61	54.61	51.70	47.78	48.07	52.67
60	136.53	49.15	54.61	55.34	50.97	49.52	47.77	48.14

From: White Threads: 16 LLC Buffers: 48K
To: Gold TTRT: 11ms MTU: 4352 Bytes

TABLE 78: TWO PROCESSORS, 7TH TEST RESULTS

Window Size (K bytes)	File A Mbps	File B Mbps	File C Mbps	File D Mbps	File E Mbps	File F Mbps	File G Mbps	File H Mbps
4	32.77	32.77	32.77	33.55	32.16	32.13	32.27	32.19
12	30.04	32.77	29.49	29.73	30.34	31.85	31.97	31.41
20	32.77	60.07	47.33	53.16	49.52	51.63	52.06	50.21
28	87.99	54.61	47.33	54.61	49.52	53.40	51.22	52.25
36	425.98	49.15	47.33	54.61	50.97	55.34	51.18	52.71
44	32.77	43.69	50.97	53.16	50.97	54.37	53.39	51.84
52	209.09	60.07	54.61	52.43	49.52	47.23	50.47	49.44
60	32.77	60.07	54.61	56.80	48.48	48.03	43.29	44.33

From: White Threads: 8 LLC Buffers: 48K
To: Gold TTRT: 25ms MTU: 4352 Bytes

TABLE 79: TWO PROCESSORS, 8TH TEST RESULTS

Window Size (K bytes)	File A Mbps	File B Mbps	File C Mbps	File D Mbps	File E Mbps	File F Mbps	File G Mbps	File H Mbps
4	30.04	32.77	32.77	34.33	32.77	31.16	31.94	31.86
12	32.77	30.95	33.50	32.16	31.73	30.59	31.49	31.87
20	222.09	60.07	54.61	50.97	52.43	52.61	52.43	51.26
28	95.57	54.61	58.25	56.80	54.61	51.81	53.85	53.10
36	32.77	43.69	50.97	58.98	53.16	50.05	52.60	54.26
44	191.75	54.61	50.97	54.61	49.93	52.78	53.47	53.40
52	32.77	60.07	50.97	53.16	45.56	50.18	49.66	53.37
60	313.12	60.07	47.33	54.61	44.52	51.02	51.26	47.16

From: White Threads: 16 LLC Buffers: 48K
To: Gold TTRT: 25ms MTU: 4352 Bytes

TABLE 80: TWO PROCESSORS, 9TH TEST RESULTS

Window Size (K bytes)	File A Mbps	File B Mbps	File C Mbps	File D Mbps	File E Mbps	File F Mbps	File G Mbps	File H Mbps
4	32.77	32.77	32.77	33.55	30.95	31.48	31.94	32.03
12	32.77	29.13	31.31	29.37	30.34	30.66	31.32	30.72
20	32.77	54.61	58.25	48.06	49.52	50.84	48.80	49.98
28	226.65	60.07	61.90	49.52	49.52	50.05	48.84	50.97
36	240.30	54.61	54.61	55.34	50.24	49.25	50.46	50.70
44	118.33	60.07	50.97	54.61	46.60	46.57	47.50	48.41
52	178.40	54.61	54.61	53.16	39.44	35.62	40.79	38.74
60	163.84	49.15	54.61	52.43	30.22	26.43	28.45	24.03

From: White Threads: 8 LLC Buffers: 56K
To: Gold TTRT: 8ms MTU: 4352 Bytes

TABLE 81: TWO PROCESSORS, 10TH TEST RESULTS

Window Size (K bytes)	File A Mbps	File B Mbps	File C Mbps	File D Mbps	File E Mbps	File F Mbps	File G Mbps	File H Mbps
4	32.77	32.77	32.77	34.33	32.16	32.13	32.27	32.79
12	32.77	29.13	31.68	31.55	32.94	31.80	32.17	31.47
20	136.53	60.07	58.25	56.80	50.97	50.84	52.93	51.27
28	862.63	60.07	61.90	52.43	53.16	53.58	52.93	52.88
36	289.45	65.54	54.61	56.80	52.43	53.40	52.98	54.04
44	32.77	60.07	50.97	56.80	45.15	51.02	53.01	52.32
52	105.33	49.15	50.97	54.61	40.05	44.35	43.05	43.66
60	118.33	49.15	47.33	56.80	35.46	36.32	30.46	30.52

From: White Threads: 16 LLC Buffers: 56K
To: Gold TTRT: 8ms MTU: 4352 Bytes

TABLE 82: TWO PROCESSORS, 11TH TEST RESULTS

Window Size (K bytes)	File A Mbps	File B Mbps	File C Mbps	File D Mbps	File E Mbps	File F Mbps	File G Mbps	File H Mbps
4	32.77	32.77	34.59	34.33	31.55	32.13	32.11	31.86
12	30.04	29.13	29.49	29.25	29.73	31.16	30.58	31.01
20	300.37	65.54	47.33	50.97	49.52	50.05	51.22	50.98
28	340.42	49.15	54.61	56.80	50.97	50.97	52.93	51.61
36	357.72	60.07	58.25	56.80	54.61	50.84	53.01	52.46
44	404.14	49.15	54.61	52.43	52.43	52.43	51.33	53.11
52	118.33	49.15	50.97	52.43	44.52	45.12	43.11	44.36
60	136.53	60.07	54.61	56.80	46.78	32.39	34.31	27.51

From: White   Threads: 8   LLC Buffers: 56K
To: Gold   TTRT: 5ms   MTU: 4352 Bytes

TABLE 83: TWO PROCESSORS, 12TH TEST RESULTS

Window Size (K bytes)	File A Mbps	File B Mbps	File C Mbps	File D Mbps	File E Mbps	File F Mbps	File G Mbps	File H Mbps
4	30.04	30.95	34.59	33.55	32.16	31.80	31.47	32.11
12	30.04	32.77	30.58	28.64	30.34	30.55	30.70	30.48
20	87.99	65.54	50.97	52.43	49.52	50.84	51.18	50.96
28	209.09	54.61	50.97	56.80	52.43	53.40	53.39	52.66
36	32.77	43.69	58.25	52.43	50.97	51.81	52.47	52.33
44	209.09	49.15	61.90	52.43	51.70	49.38	52.43	49.68
52	267.61	49.15	54.61	54.61	37.97	40.27	43.53	41.02
60	136.53	49.15	47.33	52.43	38.77	35.34	27.04	28.61

From: White   Threads: 16   LLC Buffers: 56K
To: Gold   TTRT: 5ms   MTU: 4352 Bytes

TABLE 84: TWO PROCESSORS, 13TH TEST RESULTS

Window Size (K bytes)	File A Mbps	File B Mbps	File C Mbps	File D Mbps	File E Mbps	File F Mbps	File G Mbps	File H Mbps
4	32.77	32.77	32.77	34.33	32.16	31.80	31.62	31.54
12	32.77	29.13	30.58	29.37	30.34	31.48	31.15	30.84
20	105.33	49.15	54.61	54.61	48.06	53.40	52.06	51.40
28	240.69	60.07	61.90	52.43	48.06	53.40	50.31	52.01
36	136.53	60.07	54.61	61.17	52.43	52.43	52.47	51.29
44	199.34	43.69	54.61	53.16	50.24	50.97	50.65	50.97
52	32.77	38.23	50.97	53.16	39.43	42.37	42.94	37.45
60	295.52	43.69	50.97	53.16	37.99	30.91	27.44	26.73

From: White   Threads: 8   LLC Buffers: 56K
To: Gold   TTRT: 11ms   MTU: 4352 Bytes

TABLE 85: TWO PROCESSORS, 14TH TEST RESULTS

Window Size (K bytes)	File A Mbps	File B Mbps	File C Mbps	File D Mbps	File E Mbps	File F Mbps	File G Mbps	File H Mbps
4	32.77	30.95	34.59	33.55	31.55	31.16	31.15	31.54
12	136.53	29.13	28.40	30.34	29.73	30.01	30.88	30.34
20	105.33	54.61	50.97	50.97	48.06	50.84	50.76	50.03
28	[illegible]	60.07	54.61	56.80	48.48	50.84	51.18	49.72
36	[illegible]	49.15	50.97	56.80	50.97	50.84	51.68	51.31
44	731.56	49.15	49.15	52.43	50.97	51.63	51.26	50.63
52	249.40	60.07	58.25	56.80	41.25	42.93	46.08	44.73
60	32.77	60.07	50.97	50.97	50.66	42.39	32.90	31.35

From: White Threads: 16 LLC Buffers: 56K
To: Gold TTRT: 11ms MTU: 4352 Bytes

TABLE 86: TWO PROCESSORS, 15TH TEST RESULTS

Window Size (K bytes)	File A Mbps	File B Mbps	File C Mbps	File D Mbps	File E Mbps	File F Mbps	File G Mbps	File H Mbps
4	30.04	38.23	32.77	33.55	32.16	31.80	31.46	32.03
12	32.77	36.41	29.49	29.13	31.55	30.55	29.69	30.56
20	199.34	54.61	43.69	52.43	48.06	49.25	50.14	50.95
28	340.42	54.61	47.33	52.43	50.97	52.61	52.06	50.98
36	340.42	60.07	47.33	56.80	49.52	52.43	51.60	51.11
44	136.53	43.69	50.97	50.97	49.52	49.25	49.35	51.63
52	163.84	49.15	47.33	54.61	41.87	37.63	40.21	40.91
60	136.53	49.15	54.61	52.43	32.86	30.63	28.04	26.21

From: White Threads: 8 LLC Buffers: 56K
To: Gold TTRT: 25ms MTU: 4352 Bytes

TABLE 87: TWO PROCESSORS, 16TH TEST RESULTS

Window Size (K bytes)	File A Mbps	File B Mbps	File C Mbps	File D Mbps	File E Mbps	File F Mbps	File G Mbps	File H Mbps
4	27.31	30.95	34.59	33.55	30.34	31.80	31.65	31.39
12	30.04	30.95	30.58	29.37	29.86	30.27	30.43	30.35
20	204.73	54.61	50.97	52.43	47.02	49.38	49.38	47.82
28	[illegible]	38.23	50.97	54.61	49.52	50.84	51.79	49.47
36	32.77	49.15	50.97	56.80	52.43	50.18	51.67	50.90
44	209.35	54.61	49.15	53.16	50.97	49.13	48.88	48.58
52	105.33	60.07	49.15	56.80	39.79	35.18	40.48	39.56
60	222.09	60.07	52.79	54.61	43.71	27.27	31.75	28.45

From: White Threads: 16 LLC Buffers: 56K
To: Gold TTRT: 25ms MTU: 4352 Bytes

TABLE 88: TWO PROCESSORS, 17TH TEST RESULTS

Window Size (K bytes)	File A Mbps	File B Mbps	File C Mbps	File D Mbps	File E Mbps	File F Mbps	File G Mbps	File H Mbps
4	32.77	32.77	32.77	35.11	32.33	32.13	31.77	31.46
12	27.31	30.95	29.49	30.46	30.95	31.16	30.56	30.85
20	95.57	54.61	54.61	56.80	48.06	52.43	50.08	50.35
28	875.63	49.15	54.61	52.43	48.06	50.84	50.49	49.55
36	267.61	60.07	58.25	56.80	34.88	31.68	31.52	29.14
44	105.33	60.07	43.69	50.97	16.61	16.03	15.43	13.97
52	32.77	43.69	45.51	42.29	10.42	10.39	10.46	10.47
60	202.07	29.13	26.21	28.50	15.76	9.98	8.60	8.61

From: White　　Threads: 8　　LLC Buffers: 40K
To: Gold　　TTRT: 8ms　　MTU: 4352 Bytes

TABLE 89: TWO PROCESSORS, 18TH TEST RESULTS

Window Size (K bytes)	File A Mbps	File B Mbps	File C Mbps	File D Mbps	File E Mbps	File F Mbps	File G Mbps	File H Mbps
4	32.77	32.77	34.59	33.55	30.95	31.80	31.63	31.46
12	27.31	29.13	27.31	28.64	30.95	30.37	30.14	30.93
20	143.21	49.15	47.33	56.80	49.52	50.84	50.01	51.82
28	363.18	38.23	50.97	52.43	48.06	50.05	51.64	51.79
36	32.77	49.15	58.25	54.61	35.49	37.72	33.86	34.20
44	385.93	54.61	54.61	48.06	20.25	19.47	18.16	16.44
52	137.14	47.33	41.87	46.60	15.49	12.22	13.52	12.28
60	27.31	31.86	40.78	35.54	27.72	14.27	11.82	10.84

From: White　　Threads: 16　　LLC Buffers: 40K
To: Gold　　TTRT: 8ms　　MTU: 4352 Bytes

TABLE 90: TWO PROCESSORS, 19TH TEST RESULTS

Window Size (K bytes)	File A Mbps	File B Mbps	File C Mbps	File D Mbps	File E Mbps	File F Mbps	File G Mbps	File H Mbps
4	30.04	32.77	32.77	35.11	32.77	32.13	31.63	31.62
12	32.77	29.13	27.31	29.25	32.94	30.34	29.98	30.33
20	1017.63	54.61	54.61	52.43	50.97	50.97	48.11	49.98
28	240.30	49.15	54.61	54.61	49.52	49.25	50.76	49.29
36	32.77	43.69	54.61	52.43	37.36	27.66	33.78	29.93
44	236.35	60.07	50.97	55.34	17.61	17.01	15.02	15.97
52	163.84	38.23	45.51	47.02	13.82	12.60	11.30	11.01
60	24.58	41.87	34.22	40.21	10.92	9.22	9.88	9.02

From: White　　Threads: 8　　LLC Buffers: 40K
To: Gold　　TTRT: 5ms　　MTU: 4352 Bytes

TABLE 91: TWO PROCESSORS, 20TH TEST RESULTS

Window Size (K bytes)	File A Mbps	File B Mbps	File C Mbps	File D Mbps	File E Mbps	File F Mbps	File G Mbps	File H Mbps
4	30.04	32.77	32.77	33.55	32.16	31.80	32.44	31.94
12	32.77	29.13	32.40	32.33	30.95	31.48	32.60	32.28
20	95.57	60.07	54.61	55.34	50.97	52.61	51.64	51.81
28	163.84	49.15	54.61	58.98	52.43	51.81	52.47	52.11
36	580.72	49.15	61.90	57.53	42.70	42.85	41.20	42.48
44	408.39	60.07	58.25	52.43	24.17	23.76	22.88	24.98
52	136.53	49.15	52.79	49.52	21.30	16.31	15.64	14.92
60	161.11	45.51	45.51	44.52	34.37	22.15	11.93	13.02

From: White　Threads: 16　LLC Buffers: 40K
To: Gold　TTRT: 5ms　MTU: 4352 Bytes

TABLE 92: TWO PROCESSORS, 21ST TEST RESULTS

Window Size (K bytes)	File A Mbps	File B Mbps	File C Mbps	File D Mbps	File E Mbps	File F Mbps	File G Mbps	File H Mbps
4	32.77	29.13	34.59	33.55	30.95	30.84	31.46	31.07
12	27.31	29.13	28.40	29.25	29.86	30.27	29.85	30.25
20	163.84	60.07	50.97	54.61	50.97	50.41	48.11	50.97
28	155.65	54.61	50.97	53.16	49.52	49.14	50.54	49.60
36	32.77	54.61	54.61	52.43	38.21	39.04	38.28	39.86
44	300.37	38.23	50.97	54.61	20.70	22.40	20.85	19.60
52	237.57	36.41	49.15	55.34	15.74	14.62	13.49	12.56
60	32.77	36.41	38.23	39.43	28.27	16.50	11.57	10.39

From: White　Threads: 8　LLC Buffers: 40K
To: Gold　TTRT: 11ms　MTU: 4352 Bytes

TABLE 93: TWO PROCESSORS, 22ND TEST RESULTS

Window Size (K bytes)	File A Mbps	File B Mbps	File C Mbps	File D Mbps	File E Mbps	File F Mbps	File G Mbps	File H Mbps
4	32.77	32.77	34.59	33.55	32.77	31.80	31.95	31.86
12	24.58	30.95	29.49	29.73	29.25	30.55	30.87	31.00
20	32.77	49.15	58.25	52.43	50.97	51.63	53.35	52.44
28	357.72	60.07	61.90	54.61	49.52	51.63	53.39	52.02
36	209.35	60.07	58.25	52.43	37.97	35.63	38.66	35.97
44	136.53	60.07	50.97	52.12	21.22	19.94	19.19	17.06
52	99.86	34.59	40.05	52.12	14.32	13.29	13.40	11.85
60	27.31	41.87	37.87	52.12	25.79	11.10	10.54	10.27

From: White　Threads: 16　LLC Buffers: 40K
To: Gold　TTRT: 11ms　MTU: 4352 Bytes

TABLE 94: TWO PROCESSORS, 23RD TEST RESULTS

Window Size (K bytes)	File A Mbps	File B Mbps	File C Mbps	File D Mbps	File E Mbps	File F Mbps	File G Mbps	File H Mbps
4	32.77	32.77	32.77	33.55	32.77	31.48	31.94	32.19
12	27.31	29.13	30.58	28.16	31.55	30.55	30.88	30.84
20	163.84	43.69	58.25	54.61	50.97	48.59	50.01	50.37
28	163.84	60.07	58.25	54.61	49.52	46.81	51.26	50.46
36	236.40	60.07	61.90	54.61	40.21	30.18	32.97	36.09
44	226.65	49.15	54.61	49.52	17.11	18.16	18.16	17.45
52	163.84	40.05	38.96	42.03	13.61	12.51	13.09	11.20
60	133.80	34.59	32.77	38.70	12.96	13.13	10.77	11.18

From: White   Threads: 8   LLC Buffers: 40K
To: Gold   TTRT: 25ms   MTU: 4352 Bytes

TABLE 95: TWO PROCESSORS, 24TH TEST RESULTS

Window Size (K bytes)	File A Mbps.	File B Mbps	File C Mbps	File D Mbps	File E Mbps	File F Mbps	File G Mbps	File H Mbps
4	32.77	43.69	32.77	34.33	31.55	31.48	31.46	31.38
12	30.04	27.31	30.58	30.34	30.34	30.55	29.84	30.70
20	370.05	43.69	58.25	49.62	49.52	45.58	49.93	50.96
28	99.86	60.07	50.97	39.32	35.37	31.76	33.59	31.37
36	105.33	43.69	46.97	38.35	19.35	20.65	19.89	19.76
44	22.76	37.14	34.59	24.52	10.91	11.70	10.42	9.39
52	84.65	24.76	26.21	19.58	9.58	8.59	8.83	7.65
60	22.76	26.76	18.98	18.03	8.01	8.58	7.65	6.35

From: White   Threads: 16   LLC Buffers: 40K
To: Gold   TTRT: 25ms   MTU: 4352 Bytes

TABLE 96: TWO PROCESSORS, 25TH TEST RESULTS

Window Size (K bytes)	File A Mbps	File B Mbps	File C Mbps	File D Mbps	File E Mbps	File F Mbps	File G Mbps	File H Mbps
4	32.77	32.77	34.59	37.45	35.11	36.20	36.16	36.16
12	27.31	27.31	27.31	26.79	28.76	27.93	28.74	27.94
20	209.09	43.69	54.61	52.43	49.52	50.05	50.39	50.47
28	209.35	54.61	58.25	54.61	50.97	51.63	52.52	54.02
36	313.12	54.61	54.61	52.43	50.66	53.40	51.64	53.81
44	32.77	32.77	38.23	48.06	42.29	45.18	40.85	43.08
52	62.39	24.58	31.37	33.72	27.15	34.44	28.68	25.57
60	23.67	16.12	27.03	25.75	36.43	34.19	21.90	20.40

From: White   Threads: 8   LLC Buffers: 48K
To: Gold   TTRT: 8ms   MTU: 4192 Bytes

TABLE 97: TWO PROCESSORS, 26TH TEST RESULTS

Window Size (K bytes)	File A Mbps	File B Mbps	File C Mbps	File D Mbps	File E Mbps	File F Mbps	File G Mbps	File H Mbps
4	32.77	32.77	36.41	34.33	36.67	36.20	36.37	36.58
12	27.31	25.49	24.03	25.91	28.76	27.23	28.88	28.26
20	573.44	54.61	47.33	49.52	53.16	46.04	50.88	49.85
28	411.42	54.61	47.33	49.52	50.97	50.35	52.06	54.49
36	105.33	49.15	45.51	53.16	49.52	50.10	51.60	51.79
44	32.77	36.41	40.78	44.52	45.56	41.09	42.08	41.80
52	25.49	24.32	28.61	29.94	22.45	28.66	26.59	28.23
60	15.93	25.40	28.76	21.53	15.90	18.60	18.59	18.05

From: White Threads: 16 LLC Buffers: 48K
To: Gold TTRT: 8ms MTU: 4192 Bytes

TABLE 98: TWO PROCESSORS, 27TH TEST RESULTS

Window Size (K bytes)	File A Mbps	File B Mbps	File C Mbps	File D Mbps	File E Mbps	File F Mbps	File G Mbps	File H Mbps
4	32.77	43.69	36.41	37.45	33.55	36.20	35.96	36.06
12	27.31	25.49	25.49	25.91	29.25	30.59	29.28	29.70
20	32.77	54.61	50.97	52.43	50.97	52.61	51.60	51.63
28	115.60	54.61	52.79	49.93	52.43	54.55	52.98	54.73
36	32.77	54.61	52.43	43.74	44.18	49.62	46.53	45.11
44	161.11	24.58	37.18	34.74	34.59	36.42	33.22	30.57
52	106.04	25.49	18.70	28.65	27.51	26.02	23.55	21.87
60	23.21	15.58	16.90	24.43	20.91	18.08	17.50	13.09

From: White Threads: 8 LLC Buffers: 48K
To: Gold TTRT: 5ms MTU: 4192 Bytes

TABLE 99: TWO PROCESSORS, 28TH TEST RESULTS

Window Size (K bytes)	File A Mbps	File B Mbps	File C Mbps	File D Mbps	File E Mbps	File F Mbps	File G Mbps	File H Mbps
4	32.77	43.69	34.59	37.45	35.89	35.78	36.19	35.67
12	27.31	25.49	25.49	25.02	29.73	29.16	28.40	28.50
20	118.33	60.07	54.61	50.97	49.52	50.84	51.60	52.23
28	32.77	49.15	54.61	54.61	52.43	51.81	52.06	52.89
36	209.09	49.15	50.97	50.97	49.52	51.15	52.98	50.41
44	372.93	49.15	47.33	50.97	41.68	46.47	42.04	44.85
52	23.67	36.41	38.23	36.90	31.12	28.00	28.46	27.17
60	24.58	21.30	19.93	26.06	28.48	22.34	20.34	19.24

From: White Threads: 16 LLC Buffers: 48K
To: Gold TTRT: 5ms MTU: 4192 Bytes

TABLE 100: TWO PROCESSORS, 29TH TEST RESULTS

Window Size (K bytes)	File A Mbps	File B Mbps	File C Mbps	File D Mbps	File E Mbps	File F Mbps	File G Mbps	File H Mbps
4	30.04	43.69	36.41	36.67	35.89	35.37	35.96	35.95
12	27.31	25.49	26.58	25.57	29.37	28.26	27.75	29.15
20	163.84	60.07	47.33	50.97	48.06	48.72	50.87	52.02
28	87.99	54.61	54.61	49.52	49.52	49.25	51.00	53.91
36	32.77	49.15	49.15	53.16	49.52	49.25	52.61	53.38
44	233.62	32.77	40.05	42.03	39.86	42.85	43.83	44.70
52	27.31	28.22	30.58	34.05	27.88	26.02	28.30	30.98
60	14.11	20.21	25.30	22.95	19.72	20.17	25.32	18.62

From: White   Threads: 8   LLC Buffers: 48K
To: Gold   TTRT: 11ms   MTU: 4192 Bytes

TABLE 101: TWO PROCESSORS, 30TH TEST RESULTS

Window Size (K bytes)	File A Mbps	File B Mbps	File C Mbps	File D Mbps	File E Mbps	File F Mbps	File G Mbps	File H Mbps
4	72.82	32.77	40.05	36.67	34.33	35.37	36.37	36.47
12	24.58	27.31	26.21	25.82	28.64	29.07	29.02	28.77
20	144.73	54.61	47.33	52.43	49.52	50.35	52.01	52.25
28	87.99	54.61	50.97	49.52	53.16	51.63	54.27	55.44
36	294.91	49.15	41.87	49.20	49.93	53.58	54.27	52.24
44	92.84	43.69	36.04	40.15	40.64	45.91	45.30	42.43
52	23.67	27.31	21.20	35.08	26.25	34.25	32.36	28.54
60	22.76	19.48	22.44	20.04	16.91	21.30	17.75	17.78

From: White   Threads: 16   LLC Buffers: 48K
To: Gold   TTRT: 11ms   MTU: 4192 Bytes

TABLE 102: TWO PROCESSORS, 31ST TEST RESULTS

Window Size (K bytes)	File A Mbps	File B Mbps	File C Mbps	File D Mbps	File E Mbps	File F Mbps	File G Mbps	File H Mbps
4	32.77	43.69	38.23	36.67	36.67	35.78	35.96	36.37
12	24.58	29.13	26.58	25.82	27.19	28.96	28.90	29.14
20	118.33	54.61	50.97	52.43	50.97	50.18	52.43	50.83
28	406.87	49.15	50.97	52.43	52.43	52.78	54.31	52.50
36	163.84	38.23	54.61	53.16	50.24	55.52	52.19	54.73
44	118.33	41.87	40.05	45.56	45.56	48.46	44.18	46.00
52	155.65	26.40	32.77	33.11	31.85	36.02	31.40	30.01
60	19.11	37.14	25.91	21.77	36.17	27.44	19.58	21.12

From: White   Threads: 16   LLC Buffers: 48K
To: Gold   TTRT: 5ms   MTU: 4192 Bytes

TABLE 103: TWO PROCESSORS, 32ND TEST RESULTS

Window Size (K bytes)	File A Mbps	File B Mbps	File C Mbps	File D Mbps	File E Mbps	File F Mbps	File G Mbps	File H Mbps
4	105.33	32.77	32.77	35.89	36.67	35.78	35.76	35.75
12	27.31	27.31	24.76	26.79	28.76	28.39	29.19	29.01
20	163.84	60.07	58.25	49.52	52.43	50.84	52.43	51.26
28	118.33	60.07	58.25	52.43	52.43	51.81	52.52	52.29
36	423.25	43.69	58.25	50.97	50.97	53.40	50.85	52.66
44	27.31	43.69	43.69	46.60	41.61	19	40.60	43.53
52	105.59	27.31	28.40	28.45	35.32	32.74	27.23	29.29
60	198.88	21.85	22.05	23.14	28.05	25.07	18.12	21.63

From: White Threads: 16 LLC Buffers: 48K
To: Gold TTRT: 25ms MTU: 4192 Bytes

TABLE 104: TWO PROCESSORS, 33RD TEST RESULTS

Window Size (K bytes)	File A Mbps	File B Mbps	File C Mbps	File D Mbps	File E Mbps	File F Mbps	File G Mbps	File H Mbps
4	173.08	38.23	34.59	36.67	36.67	35.78	35.96	36.80
12	27.31	25.49	24.76	25.91	29.46	28.90	29.30	28.88
20	95.57	49.15	54.61	52.43	49.52	51.63	50.80	52.27
28	118.33	60.07	54.61	52.43	50.97	53.58	52.93	53.12
36	209.35	54.61	50.97	50.97	56.80	55.34	55.75	54.55
44	209.09	49.15	50.97	49.52	50.97	50.84	52.67	54.52
52	32.77	49.15	50.97	49.52	45.15	45.12	45.94	47.08
60	23.67	38.23	37.14	42.65	40.99	37.17	39.92	35.78

From: White Threads: 8 LLC Buffers: 56K
To: Gold TTRT: 8ms MTU: 4192 Bytes

TABLE 105: TWO PROCESSORS, 34TH TEST RESULTS

Window Size (K bytes)	File A Mbps	File B Mbps	File C Mbps	File D Mbps	File E Mbps	File F Mbps	File G Mbps	File H Mbps
4	32.77	49.15	40.05	36.67	35.89	36.62	36.59	36.06
12	30.04	30.95	26.21	25.91	28.64	28.67	28.70	28.35
20	105.33	60.07	52.79	50.97	52.43	52.43	52.06	52.23
28	337.09	60.07	50.97	52.43	53.16	52.61	54.27	53.85
36	199.34	54.61	50.97	50.97	53.16	52.43	54.27	53.63
44	105.33	49.15	50.97	50.97	50.97	51.63	50.39	52.63
52	178.40	36.41	41.87	49.52	41.61	47.13	48.51	48.04
60	133.80	32.77	35.32	39.79	45.20	46.24	37.52	34.26

From: White Threads: 16 LLC Buffers: 56K
To: Gold TTRT: 8ms MTU: 4192 Bytes

TABLE 106: TWO PROCESSORS, 35TH TEST RESULTS

Window Size (K bytes)	File A Mbps	File B Mbps	File C Mbps	File D Mbps	File E Mbps	File F Mbps	File G Mbps	File H Mbps
4	32.77	32.77	36.41	35.11	36.67	34.95	35.57	36.06
12	27.31	23.67	24.76	25.82	29.73	29.44	29.54	28.82
20	105.33	49.15	50.97	50.97	51.70	51.63	51.26	51.38
28	704.51	54.61	54.61	52.43	52.43	53.58	52.89	53.60
36	330.41	54.61	50.97	50.97	51.70	53.58	54.27	55.27
44	219.06	60.07	54.61	50.97	51.70	53.58	52.52	52.41
52	101.68	54.61	41.87	45.15	45.20	47.86	50.54	48.87
60	168.13	52.79	36.41	42.36	41.75	38.63	38.67	39.02

From: White Threads: 8 LLC Buffers: 56K
To: Gold TTRT: 5ms MTU: 4192 Bytes

TABLE 107: TWO PROCESSORS, 36TH TEST RESULTS

Window Size (K bytes)	File A Mbps	File B Mbps	File C Mbps	File D Mbps	File E Mbps	File F Mbps	File G Mbps	File H Mbps
4	32.77	43.69	36.41	36.67	35.89	35.37	35.56	36.47
12	21.85	25.49	25.49	26.70	28.64	28.26	28.49	28.94
20	87.99	38.23	50.97	52.43	48.48	47.13	52.47	51.80
28	158.38	43.69	50.97	49.52	50.97	49.93	51.22	53.56
36	327.38	43.69	43.69	49.52	50.97	49.93	52.14	52.75
44	32.77	54.61	50.97	49.52	51.70	50.97	51.22	52.16
52	30.04	41.87	49.15	45.63	43.74	44.65	46.88	49.18
60	23.67	34.59	38.23	34.05	38.72	39.83	42.78	36.43

From: White Threads: 16 LLC Buffers: 56K
To: Gold TTRT: 5ms MTU: 4192 Bytes

TABLE 108: TWO PROCESSORS, 37TH TEST RESULTS

Window Size (K bytes)	File A Mbps	File B Mbps	File C Mbps	File D Mbps	File E Mbps	File F Mbps	File G Mbps	File H Mbps
4	30.04	38.23	38.23	35.89	35.89	35.37	36.16	36.47
12	30.04	27.31	27.31	25.91	28.16	27.72	27.61	28.12
20	185.69	49.15	49.15	48.06	47.02	48.82	49.89	49.57
28	32.77	49.15	47.33	49.52	48.06	50.97	52.93	50.38
36	245.90	49.15	43.69	45.98	50.24	50.84	53.35	50.26
44	87.99	54.61	45.51	49.52	45.56	48.96	51.26	49.38
52	144.99	49.15	43.69	38.40	41.92	39.66	42.78	40.30
60	114.69	28.22	29.34	27.95	34.41	32.18	30.40	30.66

From: White Threads: 8 LLC Buffers: 56K
To: Gold TTRT: 11ms MTU: 4192 Bytes

TABLE 109: TWO PROCESSORS, 38TH TEST RESULTS

Window Size (K bytes)	File A Mbps	File B Mbps	File C Mbps	File D Mbps	File E Mbps	File F Mbps	File G Mbps	File H Mbps
4	149.51	32.77	36.41	35.89	35.89	36.20	36.39	36.48
12	30.04	21.85	25.49	25.42	28.16	28.56	28.78	28.87
20	105.33	43.69	50.97	50.97	50.97	51.81	51.60	52.24
28	32.77	43.69	54.61	50.97	52.43	53.58	54.36	52.38
36	32.77	60.07	58.25	52.43	50.97	53.40	53.49	52.95
44	502.44	65.54	54.61	52.12	52.43	52.61	52.93	53.40
52	118.33	60.07	43.69	46.60	47.02	51.81	50.08	46.42
60	27.31	52.79	32.40	41.92	44.83	40.43	38.40	35.53

From: White Threads: 16 LLC Buffers: 56K
To: Gold TTRT: 11ms MTU: 4192 Bytes

TABLE 110: TWO PROCESSORS, 39TH TEST RESULTS

Window Size (K bytes)	File A Mbps	File B Mbps	File C Mbps	File D Mbps	File E Mbps	File F Mbps	File G Mbps	File H Mbps
4	30.04	32.77	34.59	36.67	35.89	36.20	35.35	35.75
12	30.04	27.31	24.76	26.70	28.28	30.01	29.03	28.99
20	136.53	54.61	54.61	48.06	50.97	51.63	50.84	51.26
28	105.33	49.15	54.61	50.97	52.43	50.05	53.39	54.07
36	87.99	54.61	50.97	50.97	52.43	51.81	53.81	54.97
44	355.59	60.07	50.97	48.06	48.06	50.84	53.39	53.11
52	85.26	60.07	43.69	46.60	41.32	46.09	50.95	49.18
60	112.87	30.95	38.23	38.07	48.48	37.84	41.38	36.57

From: White Threads: 8 LLC Buffers: 56K
To: Gold TTRT: 25ms MTU: 4192 Bytes

TABLE 111: TWO PROCESSORS, 40TH TEST RESULTS

Window Size (K bytes)	File A Mbps	File B Mbps	File C Mbps	File D Mbps	File E Mbps	File F Mbps	File G Mbps	File H Mbps
4	32.77	38.23	34.59	37.45	37.45	35.37	36.60	36.58
12	21.85	29.13	24.03	27.67	27.67	28.16	28.23	28.48
20	32.77	54.61	50.97	48.06	48.06	51.63	50.87	52.44
28	209.35	49.15	54.61	50.97	52.43	53.40	52.47	52.92
36	190.89	38.23	54.61	50.97	51.70	50.84	53.39	52.44
44	353.17	38.23	50.97	52.43	49.20	50.84	52.52	55.20
52	32.77	60.07	49.15	44.52	45.15	45.35	46.79	48.13
60	27.31	32.77	32.98	40.57	44.52	35.98	39.81	35.67

From: White Threads: 16 LLC Buffers: 56K
To: Gold TTRT: 25ms MTU: 4192 Bytes

TABLE 112: TWO PROCESSORS, 41ST TEST RESULTS

Window Size (K bytes)	File A Mbps	File B Mbps	File C Mbps	File D Mbps	File E Mbps	File F Mbps	File G Mbps	File H Mbps
4	32.77	38.23	34.59	35.11	36.67	36.20	36.39	36.60
12	27.31	23.67	26.94	26.30	29.25	28.93	29.74	28.92
20	363.18	43.69	47.33	50.97	46.60	52.43	52.06	53.33
28	367.47	38.23	45.51	52.43	50.97	55.34	53.64	52.21
36	30.04	36.41	45.51	44.78	44.52	45.43	48.52	42.86
44	151.55	36.77	25.33	24.74	26.24	28.67	26.12	23.80
52	198.88	18.31	17.82	20.99	20.65	22.50	19.65	17.03
60	17.29	16.38	14.45	16.20	26.67	25.20	14.48	12.27

From: White Threads: 8 LLC Buffers: 40K
To: Gold TTRT: 8ms MTU: 4192 Bytes

TABLE 113: TWO PROCESSORS, 42ND TEST RESULTS

Window Size (K bytes)	File A Mbps	File B Mbps	File C Mbps	File D Mbps	File E Mbps	File F Mbps	File G Mbps	File H Mbps
4	32.77	38.23	34.59	35.11	36.67	36.20	36.39	36.60
12	27.31	23.67	26.94	26.30	29.25	28.93	29.74	28.92
20	363.18	43.69	47.33	50.97	46.60	52.43	52.06	53.33
28	367.47	38.23	45.51	52.43	50.97	55.34	53.64	52.21
36	30.04	36.41	45.51	44.78	44.52	45.43	48.52	42.86
44	151.55	36.77	25.33	24.74	26.24	28.67	26.12	23.80
52	198.88	18.31	17.82	20.99	20.65	22.50	19.65	17.03
60	17.29	16.38	14.45	16.20	26.67	25.20	14.48	12.27

From: White Threads: 16 LLC Buffers: 40K
To: Gold TTRT: 8ms MTU: 4192 Bytes

TABLE 114: TWO PROCESSORS, 43RD TEST RESULTS

Window Size (K bytes)	File A Mbps	File B Mbps	File C Mbps	File D Mbps	File E Mbps	File F Mbps	File G Mbps	File H Mbps
4	32.77	32.77	40.78	35.11	36.67	35.78	35.96	36.06
12	24.58	27.31	26.21	25.42	25.99	29.73	29.69	29.41
20	87.99	54.61	48.06	50.97	46.60	51.63	52.89	51.26
28	398.68	54.61	49.15	50.97	46.60	51.94	53.81	51.61
36	841.05	49.15	47.33	35.49	38.82	45.91	45.43	41.45
44	20.94	29.13	29.86	32.02	26.63	24.80	27.30	25.61
52	20.94	28.22	24.09	20.07	20.49	20.55	18.29	16.41
60	20.94	22.76	15.62	16.84	14.03	17.62	13.67	12.74

From: White Threads: 8 LLC Buffers: 40K
To: Gold TTRT: 5ms MTU: 4192 Bytes

TABLE 115: TWO PROCESSORS, 44TH TEST RESULTS

Window Size (K bytes)	File A Mbps	File B Mbps	File C Mbps	File D Mbps	File E Mbps	File F Mbps	File G Mbps	File H Mbps
4	30.04	38.23	34.59	30.56	34.33	36.20	36.37	36.07
12	32.77	29.13	26.21	21.96	27.67	27.62	28.26	28.93
20	336.33	49.15	44.42	41.69	47.70	44.64	48.24	46.53
28	22.39	38.59	29.09	22.83	37.34	37.23	39.07	36.02
36	14.07	22.76	18.37	13.99	20.04	20.71	17.11	17.06
44	9.71	11.74	15.23	10.09	10.28	12.13	12.24	10.59
52	8.60	6.40	9.25	9.83	7.40	8.81	8.37	8.24
60	6.75	8.45	8.67	9.16	7.18	9.01	6.95	8.27

From: White   Threads: 16   LLC Buffers: 40K
To: Gold   TTRT: 5ms   MTU: 4192 Bytes

TABLE 116: TWO PROCESSORS, 45TH TEST RESULTS

Window Size (K bytes)	File A Mbps	File B Mbps	File C Mbps	File D Mbps	File E Mbps	File F Mbps	File G Mbps	File H Mbps
4	32.77	38.23	34.59	36.67	36.67	35.37	35.35	35.75
12	21.85	29.13	28.40	25.02	29.13	29.44	28.52	29.20
20	209.35	54.61	47.33	50.97	49.52	52.61	52.01	50.79
28	72.82	43.69	50.97	52.43	52.43	52.61	53.81	53.35
36	30.04	32.77	38.96	41.84	45.20	46.01	47.17	45.76
44	209.35	26.40	31.68	29.63	26.39	28.94	27.35	24.47
52	23.67	24.94	26.42	24.97	20.58	18.17	19.59	20.01
60	23.67	18.20	19.70	18.59	21.62	22.31	15.22	14.55

From: White   Threads: 8   LLC Buffers: 40K
To: Gold   TTRT: 11ms   MTU: 4192 Bytes

TABLE 117: TWO PROCESSORS, 46TH TEST RESULTS

Window Size (K bytes)	File A Mbps	File B Mbps	File C Mbps	File D Mbps	File E Mbps	File F Mbps	File G Mbps	File H Mbps
4	32.77	38.23	34.59	36.67	36.67	35.78	36.37	36.27
12	27.31	25.49	26.21	26.21	27.67	30.55	31.15	29.34
20	118.33	60.07	47.33	48.06	49.52	50.84	50.88	50.90
28	219.06	54.61	54.61	52.43	52.43	52.61	54.27	52.96
36	32.77	43.69	43.69	43.07	38.70	42.65	47.77	42.97
44	20.94	30.95	24.60	39.08	29.80	28.13	25.86	28.63
52	20.94	18.57	22.78	24.66	22.02	21.26	18.62	19.02
60	21.85	18.02	20.62	17.74	24.82	19.41	17.07	14.54

From: White   Threads: 16   LLC Buffers: 40K
To: Gold   TTRT: 11ms   MTU: 4192 Bytes

TABLE 118: TWO PROCESSORS, 47TH TEST RESULTS

Window Size (K bytes)	File A Mbps	File B Mbps	File C Mbps	File D Mbps	File E Mbps	File F Mbps	File G Mbps	File H Mbps
4	32.77	38.23	36.41	36.67	36.67	36.62	36.80	36.47
12	30.04	25.49	26.21	26.70	30.46	29.47	29.15	28.21
20	199.34	49.15	54.61	50.97	52.43	50.84	50.87	52.23
28	622.59	65.54	54.61	50.97	48.06	52.43	53.47	54.03
36	136.53	38.23	47.33	42.29	40.90	46.29	42.24	41.95
44	21.85	24.94	29.15	28.02	24.57	25.73	19.76	22.86
52	20.48	20.68	17.63	19.46	16.82	15.81	14.08	16.59
60	15.93	22.06	15.48	17.38	18.20	14.40	11.59	10.89

From: White Threads: 8 LLC Buffers: 40K
To: Gold TTRT: 25ms MTU: 4192 Bytes

TABLE 119: TWO PROCESSORS, 48TH TEST RESULTS

Window Size (K bytes)	File A Mbps	File B Mbps	File C Mbps	File D Mbps	File E Mbps	File F Mbps	File G Mbps	File H Mbps
4	32.77	38.23	34.59	35.11	36.67	36.20	36.17	36.16
12	32.77	23.67	26.58	27.67	28.16	28.90	28.75	28.30
20	118.33	54.61	58.25	54.61	49.52	50.84	51.18	51.60
28	253.69	49.15	58.25	49.52	50.97	51.81	53.39	50.76
36	30.04	30.95	37.14	36.32	40.21	41.76	37.38	39.04
44	26.40	20.39	24.88	21.32	24.71	26.89	24.72	22.68
52	21.85	14.93	14.40	17.64	17.74	13.89	13.84	13.94
60	18.20	12.85	13.86	15.15	18.12	12.20	13.85	12.18

From: White Threads: 16 LLC Buffers: 40K
To: Gold TTRT: 25ms MTU: 4192 Bytes

TABLE 120: TWO PROCESSORS, 49TH TEST RESULTS

Window Size (K bytes)	File A Mbps	File B Mbps	File C Mbps	File D Mbps	File E Mbps	File F Mbps	File G Mbps	File H Mbps
4	32.77	32.77	32.77	33.55	31.55	32.77	31.95	31.86
12	30.04	29.13	31.68	31.55	31.55	31.84	31.46	31.55
20	249.40	60.07	54.61	52.43	50.97	52.43	50.14	49.30
28	209.35	65.54	54.61	52.43	52.43	52.61	52.93	51.52
36	32.77	54.61	58.25	52.43	54.61	54.37	52.47	51.58
44	190.89	60.07	54.61	53.16	50.97	52.61	52.47	51.84
52	372.93	38.23	50.97	52.43	42.65	45.78	44.43	40.78
60	136.53	43.69	47.33	51.70	45.56	36.85	37.14	29.40

From: White Threads: 8 LLC Buffers: 48K Max Throughput Prediction Test
To: Gold TTRT: 8ms MTU: 4352 Bytes

TABLE 121: TWO PROCESSORS, 50TH TEST RESULTS

Window Size (K bytes)	File A Mbps	File B Mbps	File C Mbps	File D Mbps	File E Mbps	File F Mbps	File G Mbps	File H Mbps
4	32.77	30.95	32.77	32.77	34.33	34.22	34.58	34.38
12	30.04	32.77	29.49	29.86	29.73	29.44	30.56	30.48
20	235.07	60.07	45.51	48.06	46.60	43.23	43.40	44.63
28	367.47	38.23	47.33	48.06	48.06	43.69	45.30	46.27
36	270.83	43.69	43.69	49.52	37.97	34.24	33.96	35.87
44	260.58	60.07	43.69	46.60	17.51	15.19	15.35	17.34
52	176.76	49.15	43.69	44.52	8.62	7.62	7.87	8.10
60	209.35	35.50	33.86	34.59	6.45	6.15	5.99	6.27

From: Gold-50MHz Threads: 8 LLC Buffers: 48K
To: White-50MHz TTRT: 8ms MTU: 4352 Bytes

TABLE 122: TWO PROCESSORS, 51ST TEST RESULTS

Window Size (K bytes)	File A Mbps	File B Mbps	File C Mbps	File D Mbps	File E Mbps	File F Mbps	File G Mbps	File H Mbps
4	32.77	32.77	34.59	35.89	34.33	34.95	34.58	34.86
12	30.04	30.95	30.58	30.95	29.25	30.59	30.13	30.05
20	264.57	54.61	50.97	48.06	42.65	43.79	44.64	45.13
28	105.33	54.61	54.61	50.97	47.02	45.78	46.63	46.44
36	32.77	54.61	47.33	46.60	33.89	33.87	33.01	33.48
44	32.77	49.15	47.33	52.43	13.10	15.06	13.20	13.21
52	163.84	41.87	43.69	40.83	7.82	6.95	7.32	7.05
60	27.31	29.13	29.34	29.75	5.36	5.87	5.80	5.81

From: White-50MHz Threads: 8 LLC Buffers: 48K
To: Gold-50MHz TTRT: 8ms MTU: 4352 Bytes

APPENDIX F: GLOSSARY OF TERMS

802.2	IEEE standard for the Logical Link Control.
ACK	Acknowledge. A network packet acknowledging the receipt of data.
ARP	Address Resolution Protocol. A TCP/IP protocol to translate an IP address into a MAC address.
ANSI	American National Standards Institute. A private organization that coordinates some United states standards-making. Represents the United States to the International Standards Organization.
ARPA	Advanced Research Projects Agency. A Department of Defense agency that has helped fund many computer projects including ARPANET, the Berkeley version of Unix and TCP/IP. ARPA use to be known as DARPA.
ARPANET	Advanced Research Projects Agency Network. A Department of Defense sponsored network of military and research organizations. Replaced by the Defense Data Network (DDN).
ASIC	Application-Specific Integrated Circuits.
asynchronous	FDDI term for data transmission where all requests for service contend for a pool of ring bandwidth.
bandwidth	The amount of data that can be moved through a particular communications link. FDDI has a bandwidth of 100 Mb/s.
beacon	A token ring packet that signals a serious failure on the ring.
BER	Bit Error Rate.
bps	Bits per second. Transmission speed over some media.
CCITT	*Comite Consultatif International Telegraphiqes et Telephonique* (Consultative Committee for International Telephone and Telegraph). Standards-making body administered by the International Telecommunications Union.
DARPA	Defense Advanced Research Projects Agency. See ARPA.

DAS — Dual Attached Stations. FDDI term for a node that is attached to both the primary and seco⁻dary fiber optic cables (as opposed to a node that is connected tc ıe ring via a concentrator or not dual attached.

DDN — Defense Data Network. A network for the Department of Defense and their contractors based on the TCP/IP and X.25 networking protocols.

DLL

DMA — Direct Memory Access. This is a device (controller) for controlling the tr.nsfer of data directly to or from the memory without invol g the processor. The DMA controller becomes the bus master and directs the reads or writes between itself and memory.

DNS — Domain Name System. A mechanism used in the Internet for translating names of host computers into addresses. The DNS also allows host computers not directly on the Internet to have registered names in the same style.

FDDI — Fiber Distributed Data Interface. A 100 M/bs fiber optic LAN standard based on the token ring.

FTP — File Transfer Protocol. FTP is the Internet standard for file transfer. FTP was designed from the start to work between different hosts, runing different operating systems and using different file structures. RFC 959 is the official specification for FTP.

ICMP — Internet Control Message Protocol. ICMP is often considered part of the IP layer. It communicates error messages and other conditions that require attention. ICMP messages are transmitted within IP datagrams. RFC 792 contains the official specification of ICMP.

IEEE — Institute of Electronic and Electrical Engineers. A leading standard-making body in the United States, responsible for the 802 standards for local area networks.

IGMP — Internet Group Management Protocol. IGMP lets all the systems on a physical network know which hosts currently belong to which multicast groups. This information is required by the multicast routers, so they know which multicast datagrams to forward onto which interfaces. IGMP is defined in FRC 1112.

Internet	A collection of networks that share the same namespace and use the TCP/IP protocols.
IP	Internet Protocol. The network layer protocol for the Internet.
ISO	International Standards Organization.
LAN	Local area network. Usually refers to Ethernet or token ring networks.
LLC	Logical Link Control. The upper portion of the data link layer, defined in the IEEE 802.2 standard. The logical link control layer presents a uniform interface to the user of the data link service, usually a network layer. Underneath the LLC sublayer of the data link layer is a Media Access Control (MAC) sublayer. The MAC sublayer is responsible for taking a packet of data from the LLC and submitting it to the particular data link being used.
MAC	Media Access Control. This layer provides fair and deterministic access to the medium.
Mbps	Million bits per second. 2^{20} bits of information (usually used to express a data transfer rate; as in, 1 megabit/second - 1 Mbps).
MTU	Maximum transfer unit. The biggest piece of data that can be transferred by the data link layer.
NAK	Negative acknowledgment. Response to nonreceipt or receipt of a corrupt packet of information.
NFS	Network File System. A distributed file system developed by Sun Microsystems and widely used on TCP/IP systems.
NIS	Network Information Service. Name service in the Sun Open Network Computing (ONC) family.
NPI	Network Peripheral Inc. The manufacture of the FDDI interface cards used in this investigation on the Sun SPARC workstations.
NRZI	Nonreturn-to-Zero Inverted. NRZI is an example of differential encoding. In differential encoding, the signal is decoded by comparing the polarity of adjacent signal elements rather than determining the absolute value of a signal element.
OSI	Open System Interconnection.
PCM	Physical Connection Management.

PHY	Physical Layer. PHY provides the media independent functions associated with the OSI physical layer.
PMD	Physical Medium Dependent Layer. PMD specifies the transmitters, receivers and other associated hardware
PROM	Programmable Read-Only Memory.
RARP	Reverse Address Resolution Protocol.
RISC	Reduced Instruction Set Computer. Generic name for CPUs that use a simpler instruction set than more tradi: onal designs. The Sun SPARC workstation uses RISC technology.
SMT	Station Management document. This layer provides the capability to monitor the FDDI network. SMT can provide services such as node initialization, bypassing faulty nodes and recovery.
SPARC	Scalable Processor Architecture. A reduced instruction set (RISC) processor developed by Sun and licensed by several vendors including AT&T and Texas Instruments.
SUN	Stanford University Network. This name was given for a printed circuit board developed in 1981 that was designed to run the UNXI operating system.
TCP/IP	Transmission Control Protocol/Internet Protocol. This is a common shorthand which refers to the suite of application and transport protocols which run over IP. These include FTP, Telnet, SMTP, and UDP.
THT	Token holding timer. Token ring and FDDI term for the amount of time a node can transmit data before sending the token back out to the ring.
TTRT	Target token rotation time. A term used in FDDI to set performance parameters. The TTRT serves as a measure of expected delay and is used, among other things, to set time-out parameters.
UDP	User Datagram Protocol.

LIST OF REFERENCES

[ALBE94] Albert, B., Jayasumana, A., *FDDI and FDDI-II, Architecture, Protocols, and Performance*, Artech House, 1994

[AMD92] Advanced Micro Devices, *Multimedia Over FDDI*, Conference Paper Reprint, 1992

[COMM91] Communications Week, *FDDI Poster*, Access Media Inc, 1991

[CLAR89] Clark, D., Jacobson, V., Romkey, J. and Salwen, H. *An Analysis of TCP Processing Overhead*, IEEE Communications Magazine, June 1989

[DIGI93] Digital Equipment Corporation, *Digital's Solution to High-Speed Network Computing: FDDI EISA NICs*, White Paper, 1993

[DIGI93] Digital Equipment Corporation, *High Performance TCP/IP for OSF/1 Alpah AXP Workstaions*, White Paper, March 1993

[DRUS93] Druschel, P., Abbott, M., Pagels, M., Peterson, L., *Network Subsystem Design*, IEEE Network Magazine, July 1993

[DOD83] .Department of Defence. *Military Standard Internet Protocol*, MIL-STD-1777, August 12, 1983

[GRAY91] Gray, J., *The Benchmark Handbook*, Morgan Kaufmann Publishers, Inc., 1991

[HEAT89] Heatley, S., Stokesberry, D., *Analysis of Transport Measurements Over a Local Area Network, IEEE Communications Magazine*, June 1989

[HENN90] Hennessy, J., Patterson D, Computer Architecture A Quantitative Approach, Morgan Kaufmann Publishers Inc., 1990

[HESL93] Heslop, B., Angell, D., *Mastering Solaris 2*, Sybex, 1993

[JAIN91] Jain, R., *The Art of Computer Systems Performance Ananlysis*, John Wiley & Sons, Inc., 1991

[JAIN94] Jain, R., *FDDI Handbook, High-Speed Networking Using Fiber and Other Media*, Addison Wesley, 1994

[LANT93] LAN TIMES, *Buyers Directory Issue*, McGraw-Hill, August 1993

[MALA92] Malamud, C., *Analyzing Sun Networks*, Van Nostrand Reinhold, 1992

[MINO91] Minoli, D., *Telecommunications Technology handbook*, Artech House, Inc., 1991

[NNBM94] Neal Nelson Bench Mark, *Business Benchmark Test Descriptions*, Neal Nelson & Associates, Chicago IL, 1994

[NPI93] Network Peripherals Inc., *SBus FDDI Interface, User's Manual*, Netwok Peripherals Inc., 1993

[PATT94] Patterson, D. Hennessy, J., *Computer Organization & Design, The Hardware/Software Interface*, Morgan Kaufmann Publishers Inc, 1994

[POWE93] Powers, J., *An Introduction to Fiber Optic Systems*, Aksen Associates Inc., 1993

[SASI91] Littell, R., Freund, R., Spector, P., *SAS System for Linear Models*, SAS Institute Inc, 1991

[SILIC91] SiliconGraphics, *Iris Indigo 4D/RPC Technical Report*, SiliconGraphics 1991

[STAL91] Stallings, W., *Data and Computer Communications*, 3rd ed., Macmillan Publishing Co., 1991

[STEV94] Stevens, R., *TCP/IP Illustrated, Volume 1, The Protocols*, Addison Wesley, 1994

[SUNM90] Sun Microsystems, Inc., *SPARCstation 10 System Architecture, Technical White Paper*, 1992

INITIAL DISTRIBUTION LIST

Defense Technical Information Center Cameron Station Alexandria, VA 22304-6145	2
Dudley Knox Library Code 052 Naval Postgraduate School Monterey, CA 93943	2
LTC Steven A. Ruegnitz 127 IMA Det R&D 17 Muirfield Lane Bridgewater, NJ 08807-1269	2
Director U.S. Army Research Laboratory ATTN: AMSRL-CI (CPT Schivley) APG, MD 21005-5067	10
Chairman, Code 37 CS Computer Science Department Naval Postgraduate School Monterey, CA 93943	2
Professor G.M. Lundy, Code CS/Ln Computer Science Department Naval Postgraduate School Monterey, CA 93943-5000	2
Professor Amar Zaky, Code CS/Za Computer Science Department Naval Postgraduate School Monterey, CA 93943-5000	2
Professor Shridhar Shukla, Code EC/Sh Computer Science Department Naval Postgraduate School Monterey, CA 93943-5000	2

Sun Microsystems
Attn: Patrick Barrett 2
1842 N. Shoreline Blvd.
MS UMTV80-01
Mountain View, CA 94043-1100

Network Peripherals Inc
ATTN: Chia-ming Huang 2
1371 McCarthy Blvd
Milpitas, CA 95035

Jourdan Bailey 2
1160 Pomeroy Ave.
Santa Clara, CA 95051