(2)

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

# FDDI INSTALLATION AND PERFORMANCE ANALYSIS

by

Gifford Allen Hammar

December 1992

Thesis Advisor: Professor Luqi

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

# REPORT DOCUMENTATION PAGE

| | |
|---|---|
| 1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED | 1b. RESTRICTIVE MARKINGS |
| 2a SECURITY CLASSIFICATION AUTHORITY | 3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited |
| 2b. DECLASSIFICATION/DOWNGRADING SCHEDULE | |
| 4. PERFORMING ORGANIZATION REPORT NUMBER(S) | 5. MONITORING ORGANIZATION REPORT NUMBER(S) |
| 6a. NAME OF PERFORMING ORGANIZATION Electrical and Computer Eng Dept. Naval Postgraduate School; 6b. OFFICE SYMBOL (if applicable) EC | 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)
FDDI INSTALLATION AND PERFORMANCE ANALYSIS (U)

12. PERSONAL AUTHOR(S)
Hammar, Gifford Allen

| 13a. TYPE OF REPORT | 13b. TIME COVERED | 14. DATE OF REPORT (Year, Month, Day) | 15. PAGE COUNT |
|---|---|---|---|
| Master's Thesis | FROM 10/90 TO 09/92 | December 1992 | 196 |

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 | | |
|---|---|---|
| FIELD | GROUP | SUB-GROUP |
| | | |
| | | |

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FDDI, Ethernet, networks, high speed networks, performance, performance analysis, network protocols, IEEE 802.3 protocol, ANSI X3T9.5 protocol

19. ABSTRACT (Continue on reverse if necessary and identify by block number)
This thesis discusses the theory behind collision based and noncollision based network protocols. From this basis, a complete theoretical performance analysis is performed on both Ethernet and FDDI. The CAPSnet FDDI installation is discussed and evaluated. Actual performance tests for both Ethernet and FDDI are provided and the results are discussed in detail. The test results are compared and analyzed. Actual performance is compared to theoretical performance. An explanation is provided to explain why actual performance does not match theoretical performance.

| | | |
|---|---|---|
| 20. DISTRIBUTION/AVAILABILITY OF ABSTRACT [✓ UNCLASSIFIED/UNLIMITED] ☐ SAME AS RPT. ☐ DTIC USERS | 21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED | |
| 22a. NAME OF RESPONSIBLE INDIVIDUAL Luqi | 22b. TELEPHONE (Include Area Code) (408) 646-2735 | 22c. OFFICE SYMBOL CS/LQ |

DD FORM 1473, 84 MAR    83 APR edition may be used until exhausted. All other editions are obsolete    SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED

# *FDDI INSTALLATION AND PERFORMANCE ANALYSIS*

by

*Gifford Allen Hammar*
*LT, USCG*
*Bachelor of Science in Electrical Engineering, U.S. Coast Guard Academy, 1981*

Submitted in partial fulfillment of the
requirements for the degree of

**MASTER OF SCIENCE IN ELECTRICAL ENGINEERING**

from the

**NAVAL POSTGRADUATE SCHOOL**
December 1992

Author: [illegible]
*Gifford Allen Hammar*

Approved By: [illegible]
*Professor Luqi*, Thesis Advisor

[illegible]
*Professor Shridhar Shukla*, Associate Advisor

M. A. Morgan
*Michael A. Morgan*, Chairman,
Department of Electrical and Computer Engineering

## ABSTRACT

This thesis discusses the theory behind collision based and noncollision based network protocols. From this basis, a complete theoretical performance analysis is performed on both Ethernet and FDDI. The CAPSnet FDDI installation is discussed and evaluated. Actual performance tests for both Ethernet and FDDI are provided and the results are discussed in detail. The test results are compared and analyzed. Actual performance is compared to theoretical performance. An explanation is provided to explain why actual performance does not match theoretical performance.

## TABLE OF CONTENTS

## LIST OF TABLES

# LIST OF FIGURES

# I. INTRODUCTION

The purpose of this thesis is to explore the installation and performance of an FDDI (Fiber Distributed Data Interface) network in a laboratory environment. FDDI is a high speed, high reliability collision–free networking protocol. The motivation behind the FDDI installation was two–fold. First, CAPS (Computer Aided Prototyping System), a rapid software prototyping system designed and developed at the Naval Postgraduate School, is very software intensive. The software base contains several megabytes of reusable code. Many of the file transfers are large and, as a result, Ethernet can quickly become saturated if two or three developers are working on the system at the same time. Since FDDI is not contention based and its data transfer rate is 10 times higher than that of Ethernet, we felt that this was a good way to increase overall network throughput. Second, CAPS is aimed at reducing life–cycle costs associated with software development in large, distributed real–time systems. In order to provide meaningful feasibility and timing assessments of designs for systems of this type, the network must have a deterministic upper bound on communication times. Ethernet does not provide that capability–FDDI does.

This thesis is divided into six chapters. This chapter provides an overview and definitions, as well as comparisons to previous work in this field and expected benefits. The second chapter deals with the network protocols, both hardware and software. It also provides a very brief history of network evolution and concludes with a comparative analysis of theoretical timing limits. The third chapter details the actual hardware setup in the lab, including the servers, workstations and the concentrator. The tourth chapter discusses the FDDI installation and the concentrator configuration. The fifth chapter picks up the performance analysis thread again, but this time focuses on actual timed observations, rather than expected "best-case" results. Results from both Ethernet and FDDI observations are included. Results from commercially available modelling software are included to provide a reality check. Finally, the sixth chapter ties the actual and theoretical results together and provides ideas for further research in this area.

## A. SCOPE

The scope of this investigation was to:

- Conduct a performance analysis on original Ethernet installation
- Investigate the requirements for FDDI hardware
- Recommend an appropriate hardware solution
- Purchase and install FDDI hardware in the CAPS laboratory
- Conduct performance evaluation on the FDDI installation

This investigation did not cover:

- Performance analysis of concentrators from other vendors
- Performance analysis of network protocols other than Ethernet and FDDI
- Performance analysis of other types of processors or operating systems not installed in the CAPS lab
- Interface cards provided by other vendors
- Nonstandard or high performance protocol stacks

## B. DEFINITIONS

The following list of terms is a starting point for the rest of the thesis. The reader must be familiar with the terms below to adequately digest the remaining chapters. In many cases, they are basic, but they are also the building blocks for further discussion.

**ACK**. Acknowledgment. The method a receiver uses to tell the sender the packet was received properly.

**Address resolution**. Conversion of a software (process) address to a hardware address. Each physical connection to the network has a unique address, but several processes may exist on one host that has only one network connection.

**ARPA**. Advanced Research Projects Agency. It originally funded research that resulted in ARPANET, which became Internet and DDN (Defense Data Network). ARPA is now DARPA (Defense Advanced Research Projects Agency).

**Bridge**. Connects networks which have the same network layer, but different data link layers. It can also be used to connect two similar networks which require isolation or address checking to reduce network loading.

**Broadcast**. Data destined for all hosts on a given network. Even though the data gets to each host sequentially in almost all cases (ALOHA is an exception), all hosts receive it. A single destination address is reserved for this purpose.

**CAPS**. Computer Aided Prototyping System. CAPS implements a rapid prototyping system. It features reusable code libraries.

**Code group**. The specific sequence of five code bits representing a DLL (Data Link Layer) symbol. Also called code–cell [Ref. 6: p. 10].

**Collision–free protocol**. Protocol which is based on a strictly controlled medium access scheme, as opposed to a contention–based protocol. Each host is allotted a specific amount of time to transmit its data.

**Contention–based protocol**. Protocol which is based on two or more hosts trying to access the same medium at the same time. Hosts can randomly access the medium and send data as long as they want.

**CSMA/CD**. Carrier Sense Multiple Access with Collision Detection. A contention–based network protocol. Ethernet is one example of this standard.

**IMP**. Interface Message Processor. A dedicated computer which connects two or more network segments and possibly one or more hosts. An IMP may also perform bridge or gateway functions. IMP is usually used when discussing ARPANET or DDN.

**IPI**. Intelligent Peripheral Interface. High performance standard for external peripheral device communications (not network interfaces) including high speed buffering and caching. Maximum data transfer rates for Sun IPI controllers are 6 megabytes per second. IPI controllers can control up to eight IPI devices.

**Isochronous data**. Data which has a constant bandwidth requirement regardless of the actual data transfer requirement. A voice channel is a good example of an isochronous channel.

**LAN**. Local Area Network. A network covering a relatively small area, usually less than 10 kilometers in size. For example, the IEEE 802.3 standard specifies a maximum length of 2.5 km for Ethernet.

**MAN**. Metropolitan Area Network. A network capable of covering a metropolitan area, usually tens to hundreds of kilometers in size. FDDI falls into this category.

**MIC**. Media Interface Connector. The plug and receptacle pair that makes the physical connection between the optical fibers and the transmitter or receiver.

**Multicast**. Data destined for a specific predefined subset of all hosts on a given network.

**NAK**. Negative Acknowledgment. The method a receiver uses to tell the sender the packet was not received properly.

**Port**. Physical connection to a network. In the case of FDDI, the ports are named according to their function. A and B ports are the dual ring (trunk ring) connections which connect Dual Attach Stations (DAS's) to concentrators. M ports are Master ports on concentrators which go to an S port (Slave port) on a Single Attach Station (SAS). Port connections are strictly controlled to prevent twisted rings and other types of hardware incompatibilities.

**Protocol**. A standardized way of performing a certain task. In networking parlance, it refers to the way packets are built, machines (hosts) access the medium and a myriad of other details that allow hosts to successfully communicate with one another.

**Protocol Efficiency.** Measure of number of bits devoted to overhead versus bits devoted to data. A 2000 byte packet with only 20 bytes of overhead is 99% efficient.

**Repeater**. Connects networks and/or subnets which have the same physical layer and do not require any isolation. Usually it is used to regenerate weak signals.

**Router**. Connects networks that have the same transport layer but different network layers.

**SCSI**. Small Computer Standard Interface. A data transfer protocol optimized for multiuser computing. Sun's SCSI controllers handle up to four disks each (the standard actually supports seven) with maximum data transfer rates between 2 and 5 megabytes per second. The SCSI standard supports many types of devices, but the most common ones are hard drives, scanners and CD–ROMs.

**Throughput**. Actual number of user data bits transferred per unit time. This is a more "warts and hairs" evaluation of how fast the network actually moves data from the source to destination. It includes retransmissions due to collisions or noise as well as delays from waiting for a clear link.

**Token–passing network**. A collision–free protocol which allows the host to transmit only when it possess the token. TokenRing, TokenBus and FDDI are examples of this network type.

**Transmission Efficiency**. Actual number of usable bits (including protocol overhead) transferred per unit time divided by the maximum number of bits transferred per unit time. For example, transferring 8 Mbps on a 10 Mbps link would yield a 0.8 (or 80%) transmission efficiency.

**Unicast**. Data destined for a single host.

**UTP**. Unshielded Twisted Pair cable. A type of cable used for Ethernet 10BaseT and standard telephone installations.

**VMEbus**. Versa Module Europe computer bus. Also known as IEEE 1014 bus and IEC 821 bus. VME supports data transfer widths of 8, 16 or 32 bits and supports 24– or 32–bit address buses. It also supports data transfers between any two locations on the bus as well as multiple bus masters [Ref. 1: p. 470]. Multiple bus mastering allows more than one processor to be physically connected to the same bus and control it without damaging any of the devices. Two other comparable bus architectures are NuBus and EISA.

**WAN**. Wide Area Network. A network which spans hundreds or thousands of kilometers. These networks can be either packet–switched or circuit–switched or a combination of the two.

## C. COMPARISON TO PREVIOUS WORK

Hasan S. AlKhatib at Santa Clara University has done some theoretical analysis [Ref. 25: p. 1] of a token ring, star, token bus and SC [Santa Clara] ring topology and has found that his designs are at least as good as or better than the token ring topology [Ref. 23: p. 253]. He and his associates did not perform any actual timing experiments nor did they consider any contention–based protocols or FDDI. Their ideas about protocol bottlenecks did provide a basis for my analysis of the timing differences between Ethernet and FDDI. After all, our goal for installing FDDI was to improve throughput on the CAPSnet.

M. Cohn of Northrop Corporation discussed lightweight protocols for SAFENET II [Ref. 32: p. 151]. SAFENET II is the militarized version of FDDI. His primary area of concern was related to the eXpress Transfer Protocol (XTP) which replaces the ISO/OSI Transport and Network Layer protocols. XTP is optimized for real–time distributed environments and is intended to be implemented in hardware rather than software to afford higher throughput. He discusses services and file transfers but does not discuss performance except in general terms. He also does not discuss Ethernet.

G. M. Lundy [Ref. 26: p. 369] proposes a method for increasing the throughput of an FDDI network by allowing dual–ring installations to transmit different data on the two rings, and having the receiver "drain" the ring and send out a sub–token to allow another station to use the remaining bandwidth. As with AlKhatib's work, this technique is mainly theoretical and would have very little impact in CAPS because the majority of the workstations are Single Attach rather than Dual Attach. Further, his work is only concerned with FDDI.

L. Green performed analytic modeling and simulation on LANES II at NASA Ames Research Center [Ref. 30: p. 441]. His work, as was H. AlKhatib's, is theoretical in nature, although the simulations provide a better interactive environment. He simulated an FDDI installation with eight to 320 stations and varying frame sizes from 512 to 4096 bytes. His work indicates that total throughput increases and transmission delays decrease with a larger number of stations. He did not compare FDDI with Ethernet or compare simulated results with actual measurements.

R. Sankar and Y. Y. Yang of the University of South Florida predict the performance analysis of FDDI using CSIM simulation software [Ref. 37: pp. 328–332]. Again, this was a simulation which was not backed up by actual timing tests. At the time the paper was written, hardware was not generally available to support timing tests. One of the important results (with respect to hard real–time systems) from their study was that at 95% load, the access time was strictly upper bounded [Ref. 37: p. 331].

A. Weaver of the University of Virginia (Charlottsville) has done some timing analyses on Manufacturing Automation Protocol (MAP) [Ref. 35: p. 75],Token

Ring [Ref. 33: p. 885], Token Bus [Ref. 34: p. 1253] and SAFENET [Ref. 36: p. 87]. He has not directly compared Ethernet and FDDI, although much of his work concerns improving protocol stack performance. He is a proponent of reducing stack overhead by eliminating layers in the ISO OSI model, thereby reducing the net overhead, sometimes at the expense of functionality. He has proposed reduced stacks for Token Bus, MAP and FDDI. The reduced stack for FDDI is called XTP (eXpress Transfer Protocol) and is used in SAFENET to improve throughput.

## D. EXPECTED BENEFITS

The contribution of this thesis is to provide a better understanding of the expected performance of the FDDI implementation. Even though network speed is not critical for CAPS in a development mode, it is critical when trying to predict real–time system performance for applications developed with CAPS and executed in a distributed environment. Hard real–time systems (such as SAFENET II, which is based on FDDI [Ref. 20: pp. 7–8]) require an *a priori* upper bound on all aspects of the system, including network communications. Ethernet cannot be employed for hard real–time systems, because there is a finite possibility, no matter how small, that a message will take an infinite amount of time to move from one station to another. In other words, the network has an unbounded delivery time. FDDI has a deterministic upper bound for delivery time, as do other collision–free protocols, and can be used for a hard real–time system.

The general results will be valid across systems, but drawing specific conclusions from the data provided here will not be accurate on other implementations. For example, a host which has a dedicated protocol processor will be much faster than one which uses the main processor for that task and hands the data off to the physical layer for transmission.

# II. NETWORK PROTOCOL

Connecting computers to a common communication channel has been a topic of interest since the mid–1960's. Initially, ARPA (now DARPA) funded research to connect several hosts. Early successes have caused the scope of the network to expand to include all six continents (excluding Antarctica). Internetworking allows geographically dispersed collaborators a quick, easy way to share notes, ideas, data and computing power.

In order to accomplish this, the computers need a standard way of passing digital data between them. Protocols provide the means to support the data transfers. Network protocols allow different machines to transfer data on the same network. Several network protocols are in common use today: TCP/IP, AppleTalk, Novell Netware and MAP, to name a few. In the following pages, I will discuss some of the underlying theory and how it applies to the Ethernet and *FDDI implementations in* the CAPS lab.

## A. NETWORKING THEORY

A network is built on the assumption that a host can communicate with any other host to which it is connected. The connection can be either direct or indirect [Ref. 3: p. 111]. A direct connection is one in which the host is connected directly to another through a LAN. An indirect connection is one in which the communicating hosts are not physically connected as in the case of a packet–switched WAN where the hosts have to use two or more IMPs or gateways.

This type of environment is also commonly referred to as a distributed computing environment because, generally, there are a number of computers that have similar capabilities that share a communications medium. A distributed computing environment takes on added significance when discussing multicasting because it means that several processors can work together over a network to complete a common task. In general, then, a network is a collection of stand–alone processors that share a communication medium or channel.

The physical links for the network can be guided, unguided or a combination of the two. Guided media can either be copper (twisted wire pair, coaxial cable, etc.) or optical fibers. Unguided media consists of radio transmissions although light could be considered in this realm as well. Laser transmission spans the two types; the specific application determines whether it is guided or unguided. When used with optical fibers, lasers are guided media. When used in free space, it behaves more like unguided media because of the propagation speed and dispersion characteristics. Optical splitters can serve two or more sites simultaneously, but the link is still point–to–point.

Internet, one of the largest WANs in the world, uses a combination of guided and unguided media to transmit data. It uses leased fiber optic and coaxial cable routes as well as satellite and terrestrial microwave routes to complete circuits.

The protocols that drive the physical links can be either contention or noncontention based. The differences and the importance of each with respect to performance will be discussed later in this chapter.

The following sections, Networking Model and Networking Protocols take the general ideas expressed above and expand them into more concrete ideas. The Networking Model section describes the ISO OSI Seven Layer Model in detail so that it can provide the basis for the Networking Protocols section.

## B. NETWORKING MODEL

In order to understand the difference between the different protocols, the overall framework must first be clear. The ISO OSI Seven Layer Model provides that framework even though it postdates many of the common network protocols in use today. I will continually refer back to the Seven Layer Model in the Network Protocols section in order to provide a clear transition from one set of standards to another.

The ISO OSI Seven Layer Model (shown in Figure 2.1) is the International Standards Organization Open Systems Interconnect protocol model. The seven layer model describes a framework in which any arbitrary layer provides services to the layer above it and uses the services of the layer below it. From bottom to top, the layers are: physical layer, data link layer, network layer, transport layer, session layer, presentation layer and application layer.

Each of these layers is described in sufficient detail in the paragraphs below to support a general understanding of the model. Further details are provided by Tanenbaum [Ref. 2: pp. 9–21] and Stallings [Ref. 4: pp. 389–399]. This discussion is intended to provide a basis for the detailed protocol analysis that follows in later sections. I will discuss the model from the bottom up.

### 1. Physical Layer

The physical layer is responsible for sending data across a channel or link. The link can be analog or digital, guided or unguided, connection–oriented or connectionless. The physical layer deals with all the physical parameters for data transmissions such as bit error probabilities, duty cycles, bit duration (which is inversely proportional to the data rate), encoding techniques, signal–to–noise ratios (SNR) and mechanical connections. An example of the physical layer is a pair of modems and a phone line. The modem takes the digital data stream and converts it to an analog data stream so the medium (phone line) can handle the data. The modem also takes care of encoding multiple bits. A 2400 bps modem uses QAM (Quadrature Amplitude Modulation) to encode four bits into one symbol. An

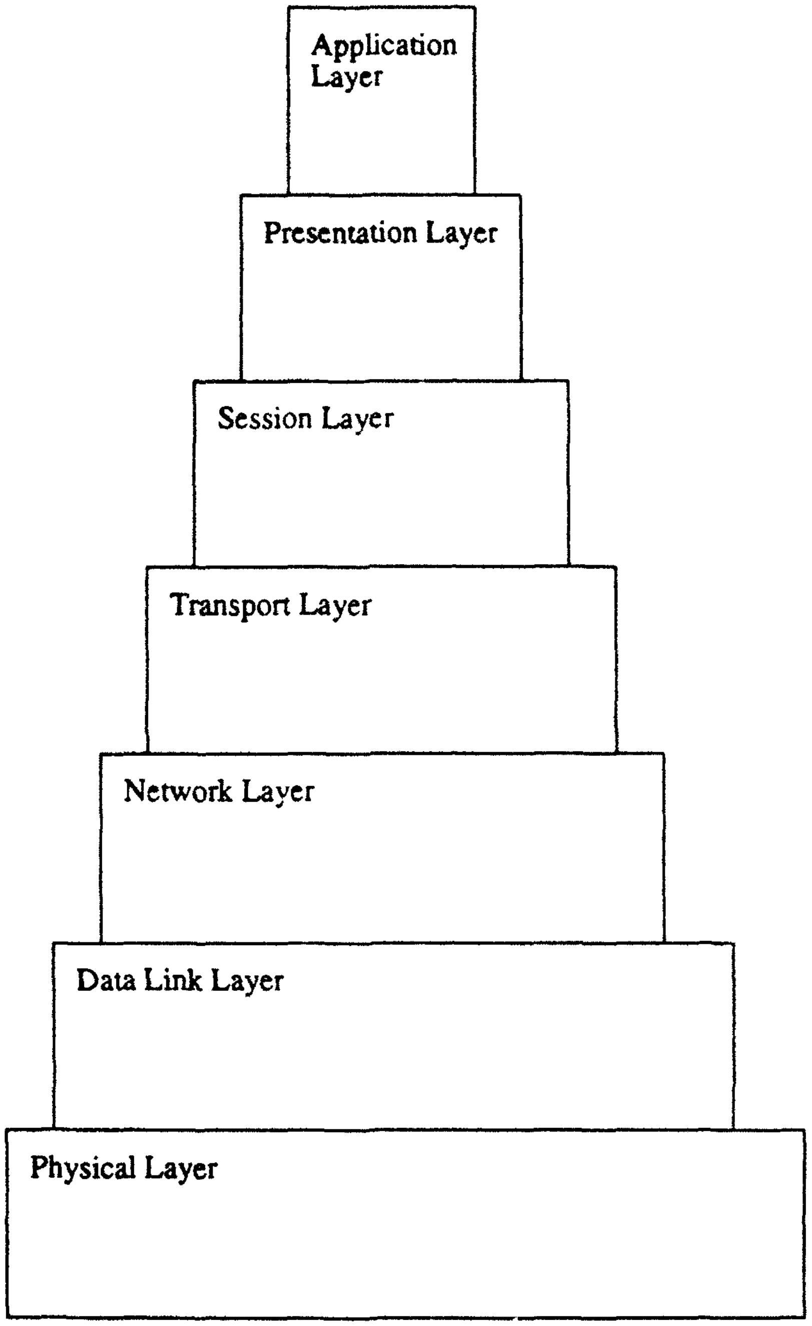


**Figure 2.1 ISO OSI Seven Layer Model**

external modem uses two standard connections: an RS–232 interface to the computer and an RJ–11 (modular phone jack) interface to the phone line. The modem handles all of the details of the way the signal appears on the phone line. A CODEC (coder/decoder) provides similar functionality for the telephone to ISDN (or other digital network) connection. The CODEC changes an analog signal to a digital signal.

### 2. Data Link Layer

The data link layer is responsible for taking a physical link that may have many errors, and making it appear to be error–free to the network layer. This is accomplished by taking a data stream and breaking it into frames or packets. The data link layer does this to accomplish two distinct missions. The first is to ensure that the network layer gets exactly one copy of each packet that the sender wants to transfer. The receiver does not want to have two or more copies of the same packet and it does not want to miss any, either. The second is to ensure that a fast transmitter does not inundate a slow receiver. This is called flow control.

### 3. Network Layer

The network layer is responsible for handling connections to other hosts. In other words, it takes care of making the connection, keeping it active and disconnecting when it is done. This also includes routing and congestion control. Congestion control is only a factor when more than one link is involved. For example, Host A wants to send a message to Host B. Hosts A and B are connected by two paths, one through Host C and the other through Host D. If Host C is busy, the network layer routes the message through Host D instead. There are several congestion control algorithms. Tanenbaum [Ref. 2: pp. 308–320] and Stallings [Ref. 4: pp. 274–280] covered them in detail.

### 4. Transport Layer

The transport layer is the lowest layer at which end–to–end communications take place. An end–to–end connection is one in which the processes at the ends of the data stream communicate even though this may occur across several gateways or routers. The lower layers talk to the host at the other end of the link. The transport layer is responsible for end–to–end error correction and flow control. It also handles the type of service (speed versus reliability issues) provided to the session layer.

### 5. Session Layer

The session layer is responsible for providing the framework for communications between applications. It also provides a means of making a connection, keeping it active and disconnecting it when completed. It is at a higher level of abstraction than the services provided by the network layer. These services are end–to–end rather than link–oriented. A remote file copy executed from the shell is an example of a task handled at the session layer.

### 6. Presentation Layer

The presentation layer accounts for differences in syntax between different types of operating systems. Representations for characters, integers, floating point numbers and other structures are stored differently on different hosts. For example, IBM mainframes generally use EBCDIC instead of ASCII notation. Negative numbers can be stored as 1's– or 2's–compliment binary structures. The receiving host has to know which representation the sending host is using in order to convert it to a meaningful display at the terminal. If data needs to be encrypted for security reasons, it usually happens in this layer since encryption is a syntactic change.

### 7. Application Layer

This is the layer where user interface usually occurs. At this layer, the notion of a real terminal and a network virtual terminal exists. In other words, there is a unique mapping between the host's notion of the user's terminal and the actual hardware that the user is operating. A good example of the application layer would be electronic mail. The user enters the destination and the text. The protocol stack takes care of converting to the proper character representation, making the connection to the recipient, routing the information most effectively, managing flow control, and taking care of error correction. The best part about all that is that the user does not have to do that manually.

### 8. Trade–offs

The preceding discussion points out one of the benefits, but also one of the biggest drawbacks of a layered protocol. The benefit is that an application that wants to use network services only needs to call the services it requires of the application layer (or other, lower layer). The disadvantage is that each layer adds its own header (and trailer in the case of the physical layer) which increases overhead. Overhead will be explored more completely in the theoretical timing analysis section.

## C. NETWORKING PROTOCOLS

There are many network protocols and each takes a different approach to provide network services. As we get into the discussion further, some of the distinctions will be confusing because of the various timeframes when the standards were proposed or approved. Some of the more popular protocols are SNA, TCP/IP, Novell Netware, AppleTalk, TOPS and Banyan Vines. SNA is the Standard Network Architecture devised and supported by IBM. Novell Netware, AppleTalk, TOPS and Banyan Vines are commercial LAN solutions. Since we only use TCP/IP in CAPSnet and the CS backbone, I will not discuss any of the others further.

The ISO OSI Seven Layer Model provides a common framework for describing network functionality and is best viewed as an umbrella because it covers not only hardware issues but software ones as well. The IEEE and ANSI standards only implement the lower layers of the ISO OSI model, while TCP, UDP and IP only implement the upper layers of the ISO OSI model. Again, the main provisions of the ISO OSI Seven Layer Model will be used as the basis for discussing TCP/IP, Ethernet and FDDI.

The first part of the discussion will center on software protocols, specifically TCP/IP, while the second part will focus on the different hardware protocols. I will discuss the derivation and evolution of the various protocols, with considerable emphasis on Ethernet and FDDI.

## 1. Software Protocols

The only software protocol suite I will discuss is TCP/IP. In our switch from Ethernet to FDDI, it remains constant. TCP/IP is the software protocol of choice for military applications.

### *a. TCP/IP*

TCP/IP stands for Transmission Control Protocol/Internet Protocol. In reality, it is a suite of protocols. TCP and IP are the two most commonly used and therefore the most widely discussed. The model which includes TCP and IP is a four layer model. From bottom to top, the layers are: network interface, internet (IP), reliable stream transport service (TCP) *or* user datagram (UDP), and application. As I did with the ISO OSI model, I will discuss each layer and the functionality it provides. I will also discuss the mapping from the TCP/IP suite to the ISO OSI Seven Layer Model which is shown in Figure 2.2.

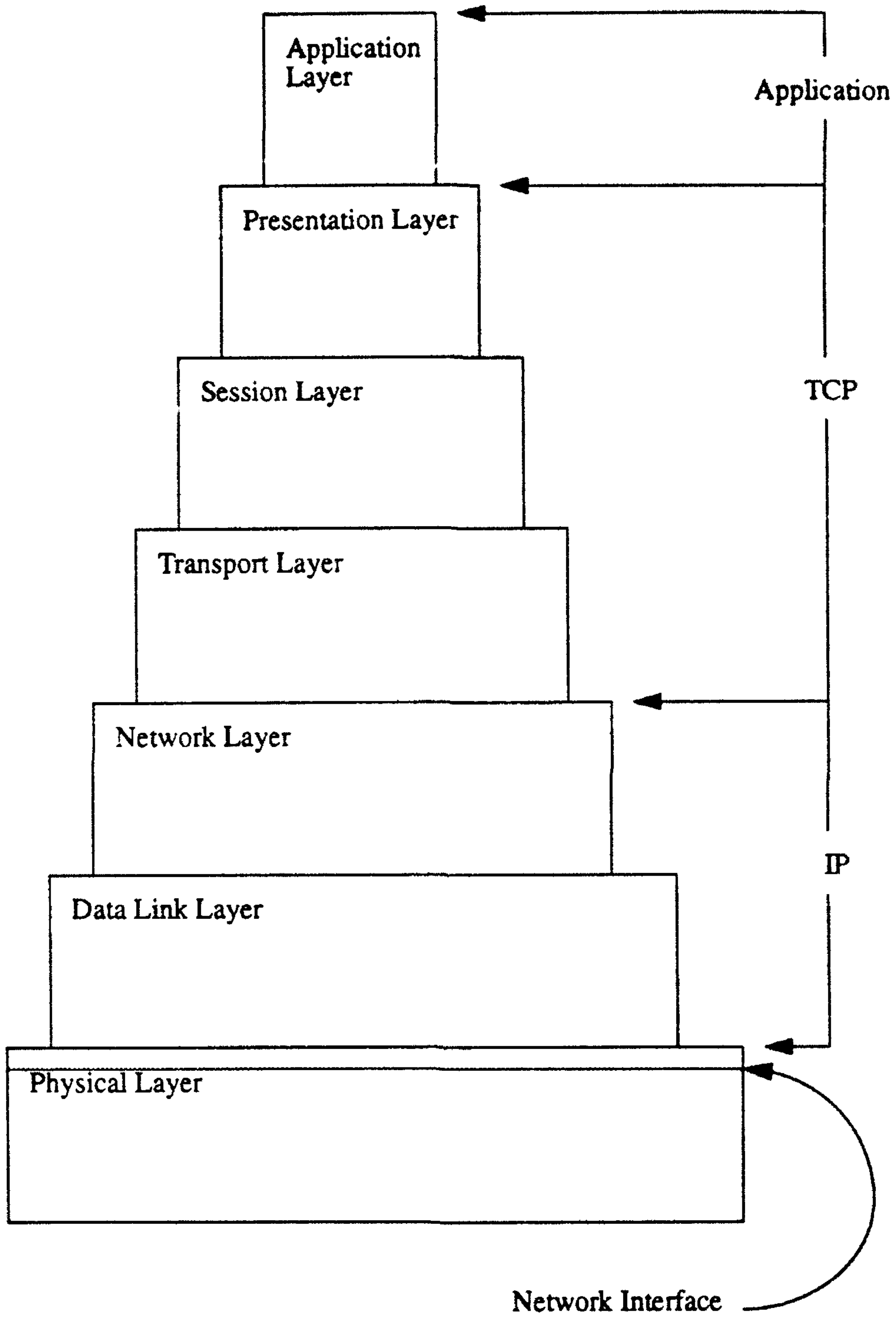


**Figure 2.2 ISO OSI model and TCP/IP**

(1) Network Interface. As with the physical layer of the ISO OSI seven layer model, this is where the host connects to the communications link. The same characteristics apply here as in that model. The network interface takes care of converting computer symbols into symbols that the link can handle. All of the physical and mechanical requirements are taken care of by the communications link. This roughly corresponds to the top portion of the Physical Layer in the ISO OSI model.

(2) Internet Protocol. The internet protocol was designed to connect many heterogeneous networks. The primary assumption was that the different networks would not have the same type of physical data transfer or similar operating systems. The IP layer is responsible for the following functions: flow control, routing, error reporting, fragmentation and congestion control [Ref. 3: pp. 111–130]. These functions roughly correspond to those provided by the ISO Data Link and Network Layers. The next few paragraphs provide the details.

IP specifies a connectionless packet delivery service [Ref. 3: p. 90] which means that the link provides unreliable, best–effort delivery service. It may send duplicate packets, lose others and deliver some (or all) out of sequence. We will see why this is not as bad as it sounds when we talk about TCP in the next section.

(a) Flow Control. IP provides link–level flow control. It is only concerned with flow control from one host or IMP to the next host or IMP, not between the hosts at the endpoints. This implementation ensures that a fast host does not overwhelm a slow one.

(b) Routing. IP handles routing for the TCP/IP suite. Functionally, routing can be divided into two different types, direct and indirect [Ref. 3: p. 111]. Direct refers to the case where the router or gateway is on the same physical network as the source or destination host or the hosts themselves are on the same physical network. Indirect routing refers to the case where the routers or gateways at the endpoints of the link are not the source or destination hosts. On a connection with *n* links, the direct cases are links 1 and *n*, while the indirect ones are *n* + 1 through *n* - 1. IP routing is normally performed with routing tables. Comer [Ref. 3: p. 113] provides details on the process. CAPSnet uses direct routing, so I will not discuss indirect routing further.

(c) Error Reporting. Error reporting is accomplished with Internet Control Message Protocol (ICMP). ICMP provides a mechanism for reporting errors back to the source, not the previous router or gateway [Ref. 3: p. 125]. ICMP is an important part of the congestion control process.

(d) Congestion Control. Congestion occurs when one router or gateway along a specific path gets more information than it can handle [Ref. 3: p. 130]. If the situation goes unchecked, the input buffers would overflow and the host would start to discard packets. This would cause senders to time-out and resend their lost packets which would further exacerbate the problem. Eventually, the network would not be able to transfer any more data and throughput would drop to zero. ICMP is used to reduce the chances of that happening. The ICMP packet can tell the source, among other things, to stop sending traffic (QUENCH) or change the route (REDIRECT) [Ref. 3: p. 127].

(e) Fragmentation. Since the model we are dealing with assumes a heterogenous network, it is reasonable to expect that the Maximum Transfer Unit (MTU) may not be the same for all of the links. If every link used Ethernet, the MTU would be 1500 bytes and the discussion would end. Not all networks use a 1500 byte MTU, though. A public packet switched network like SprintNet may use an MTU of 512 or 576 bytes. When a packet must go through a network with a smaller MTU than to one that originated it, the packet must be cut into smaller pieces, or fragmented. Fragmentation is not desirable because it adds overhead and delays the transmission process. If the routes are known in advance, IP can adjust the packet size such that the fragmentation effects are minimized. An additional problem with fragmentation is that if one fragment is damaged, the whole frame must be retransmitted, not just the damaged fragment.

(f) Framing. The IP frame consists of the following fields: VERS, HLEN, SERVICE TYPE, TOTAL LENGTH, IDENTIFICATION, FLAGS, FRAGMENT OFFSET, TIME TO LIVE, PROTOCOL, HEADER CHECKSUM, SOURCE IP ADDRESS, DESTINATION ADDRESS, IP OPTIONS, PADDING and DATA. The 4–bit VERS field contains the version of IP that created the packet. This ensures that the receiver can decode the packet. The 4–bit HLEN field specifies the header length in 32–bit words. This is necessary because the header length can vary from five to 16 words (20 to 64 bytes). The 8–bit SERVICE TYPE field contains priority information and the type of service the sender wants: high or low delay, high or low throughput, or high or low reliability. The 16–bit TOTAL LENGTH field gives the total packet length in octets (bytes). This limits the packet length to 65,536 octets including the header. The IDENTIFICATION, FLAGS and FRAGMENT OFFSET fields deal with fragmentation. The IDENTIFICATION field identifies the fragment number. The FLAGS field controls fragmentation. The FRAGMENTATION OFFSET field specifies where the fragment starts. The first fragment always starts at offset 0. Successive fragments begin at some integer multiple of the MTU–header size. The TIME TO LIVE field specifies the number of seconds a packet may be forwarded around the internet. Every gateway decrements the field and when the field reaches zero, the packet is discarded. The gateway that discards the packet sends an ICMP to the source. An expiration time is an easy way to reduce the chances of clogging the net with old traffic. The

PROTOCOL field indicates which higher level application or process created the data. The HEADER CHECKSUM field provides a 16–bit checksum for the header to allow for error detection. The SOURCE and DESTINATION ADDRESS are the 4–octet IP source and destination IP addresses, respectively. The OPTIONS field is between 0 and 44 octets long and is used for various special functions such as recording the actual route taken, specifying a route to follow and timestamping [Ref. 3: pp. 92–106]. Usually, the IP header is 20 bytes long which leaves a maximum of 65,516 bytes for data. The Sun implementation uses the MTU for the IP packet size. For Ethernet, the MTU is 1500 bytes and for FDDI, the MTU is 4478 bytes.

(3) User Datagram Protocol. The User Datagram Protocol provides unreliable connectionless service using IP [Ref. 3: p. 161]. In this respect, UDP is just a simple extension of IP. The UDP provides a simple way of determining which port on a particular host is sending the data and which port on the destination host is supposed to receive the data. Because of its simple functionality, it only needs four 16–bit fields: UDP SOURCE PORT, UDP DESTINATION PORT, UDP MESSAGE LENGTH and UDP CHECKSUM. This gives UDP an 8–byte overhead versus the 20–bytes that TCP needs. Lower overhead means faster processing. For this reason, many UNIX functions (such as `rlogin` and `cp`) use UDP as the transport protocol. I will briefly mention UDP again in the Performance Analysis chapter.

(4) Transmission Control Protocol. The Transmission Control Protocol (TCP) is properly called the Reliable Stream Transport Service. It takes the connectionless service provided by IP and makes it a reliable, end–to–end data transfer session. It roughly provides the same services that the presentation, session and transport layers provide in the ISO model. TCP exhibits five general properties: stream orientation, virtual circuit connection, buffered transfer, unstructured stream,

and full duplex connection. The roles that these properties play will be discussed in the following paragraphs.

(a) Stream Orientation. A stream orientation is best described as a First–In–First–Out (FIFO) queue. The user at the destination receives the data in the same order that the source sent it. Since TCP uses IP, which only promises to make an effort to deliver the data, how can we ensure that the users have a reliable data flow? TCP keeps track of each packet it sends and requires an acknowledgment. The most basic form of this protocol is known as stop and wait because the sender transmits a packet, then stops and waits for the destination to send an acknowledgment before it sends the next packet. If the source and destination are widely separated, this protocol is very inefficient. An incremental improvement is the Alternating Bit (AB) protocol. This protocol is better in that the sender can send one packet, pause and send the second packet. When the sender receives the first acknowledgment from the destination, it can send the next packet, and so on. Again, this protocol is not very efficient for hosts that are widely separated. The final improvement is to allow the sender to send up to *n* packets without waiting for an acknowledgment. This is known as the sliding window protocol. When the acknowledgments arrive, the sender can send more packets. From this perspective, the stop–and–wait and AB protocols are special cases of the sliding window protocol with *n* set to 1 and 2, respectively. Increasing the window size improves efficiency to a point. Maximum throughput occurs when the window size is just larger than the time it takes for one round trip. For example, if we have a 1 km Ethernet link between two hosts, the window size would be 1. The following equations show the process:

$$1526\text{bytes/packet} \cdot 8\text{bits/byte} = 12208\text{bits/packet}$$

$$(12208\text{bits})/10 \times 10^6\text{Mbps} = 1.2208 \times 10^{-3}\text{s/packet}$$

$$(2000\text{m})/2 \times 10^8\text{m/s} = 10 \times 10^{-6}\text{s}$$

$$10 \times 10^{-6}\text{s}/1.2208 \times 10^{-3}\text{s/packet} = 0.008\text{packet}$$

The Ethernet packet is 1526 bytes long and data is transmitted at a rate of 10 Mbps. Propagation speed in the cable is 2 x $10^8$ m/s. We apply the ceiling function to 0.008 to get 1. It could be larger than that and not adversely affect the network operation. If we apply the same equations to a 5000 km link over a T1 circuit (1.544 Mbps) using a 512 byte MTU, the window size is 19. This is not a very helpful heuristic when the connections can be as short as 50 m or as long as 5000 km. In a Sun server or workstation, a memory device driver stores the buffer size for the TCP sending and receiving buffers. The number of bytes dedicated to each buffer effectively determines the window size because the maximum TCP packet size is known in advance. The default buffer size is 4096 bytes for Ethernet and 24576 bytes for FDDI [Ref. 14: p. 10].

(b) Virtual Circuit Connection. The second distinguishing feature of a TCP connection is the virtual circuit. When an application on one host wants to send data to an application on another host, TCP "calls" the other host to set up the circuit. In an abstract way, this is similar to making a phone call. The difference is that the data may take several paths and the path is not dedicated to one transaction. TCP makes the connectionless IP look like a real connection.

(c) Buffered Transfer. The third distinguishing feature of a TCP connection is the buffered transfer. This means that several characters or strings may be held until TCP can build a reasonably-sized packet. When the packet is built, TCP sends it. Buffering also means that large pieces of data will be fragmented into pieces that the network can handle. A good example of this would be when one is logged into a remote host via File Transfer Protocol (FTP). After the remote host processes a file listing, the file names appear on the screen. While they are being displayed, the local host pauses at different places in the file name list. The remote host is building packets that contain a predefined number of bytes and the end of a packet will rarely occur at the end of a line of text.

(d) Unstructured Stream. The fourth distinguishing feature of a TCP connection is the unstructured stream. An unstructured stream is one which has no special formatting applied. If TCP is transferring a database data file, the transfer does not occur one record at a time. TCP takes the entire data set to be transferred and fragments it according to the MTU size. This concept is very closely related to the buffered file transfer for sending large pieces of data. In other words, there is no way to force TCP to transfer exactly one record at a time.

(e) Full Duplex Connection. The fifth distinguishing feature of a TCP connection is the full duplex connection. Full duplex means that the connection will allow two-way communications simultaneously. This feature permits the receiver to acknowledge receipt for a packet without the sender having to stop transmitting to get it.

(f) Framing. The TCP frame, like the IP frame has a maximum length of 65,536 bytes. The standard header size is 20 bytes, but it can be as large as 64 bytes. The fields for the TCP frame are: SOURCE PORT, DESTINATION PORT, SEQUENCE NUMBER, ACKNOWLEDGEMENT NUMBER, HLEN, CODE BITS, WINDOW, CHECKSUM, URGENT POINTER, OPTIONS and DATA. The 16-bit SOURCE PORT field is the process identification of the application that wants to use TCP to send data. The 16–bit DESTINATION PORT field specifies the recipient port for the data. Notice that this is not the host address, but a process on a particular host. This allows different end users (like mail, news and SQL, to name a few) to have distinct addresses and avoid data incompatibilities. Port addresses also support multicasting because several ports reside on any given host. The 32–bit SEQUENCE NUMBER field identifies the sender's pointer in the stream. It is the byte number of the first byte in the data field. The 32–bit ACKNOWLEDGEMENT NUMBER field specifies the byte number that the receiver expects to see next. This is the mechanism that allows the sliding window protocol to work properly. The 4–bit HLEN field specifies the header length in 32–bit words. This is necessary because the header length can vary from five to 16 words (20 to 64 bytes). The 6–bit CODE BITS field tells the recipient whether or not entries are valid in certain fields. Some of the functions are: urgent pointer field is valid, acknowledgment field is valid and sender has reached end of byte stream. Others are described in Comer [Ref. 3: p. 184]. The WINDOW field tells the recipient how much buffer space is available for incoming traffic. The CHECKSUM field provides a 16–bit checksum for the header to allow for error detection. The URGENT POINTER field specifies where the urgent data ends and the normal data resumes. Urgent data provides a mechanism for aborting unfinished jobs. The OPTIONS field can be from 0 to 44 bytes long and handles functions such as end–to–end flow control.

### 2. Hardware Protocols

As mentioned earlier, protocols can be either collision (or contention) based or collision–free (non–contention) based. Pure ALOHA and slotted ALOHA are examples of primitive contention protocols. Theoretical analysis showed that these protocols could be improved and "smarter" protocols were developed. These improvements can be classified as the carrier sense protocols.

The driving force behind improving the hardware protocols is to improve the transmission efficiency. The higher the transmission efficiency, the larger the percentage of the available bandwidth that is used. The goal is to approach 100% efficiency.

The following sections discuss contention protocols first, then the noncontention protocols.

### *a. Contention Protocols*

Contention protocols contend for the transmission medium. As the protocols developed, more enhancements were added to reduce the wasted bandwidth associated with collisions. I will go through the evolution of contention protocols, discussing the following protocols in order: ALOHA, slotted ALOHA, CSMA, CSMA–CD, *p*–persistent CSMA–CD and CSMA–CA.

(1) ALOHA. No network discussion would be complete without mentioning ALOHA, the network system set up at the University of Hawaii to serve its geographically remote campuses. Transmissions from the remote locations to the master (host) were on one radio frequency while the replies were on another. The protocol was very simple: if you have something to send, send it.

ALOHA pays for this simplicity in performance. Because there are two chances for a collision for each frame (once toward the beginning and once toward the end), the highest possible efficiency is 18.4%, excluding protocol overhead. By using queueing theory with Poisson distributed arrival times, the equation is:

$$S = Ge^{-2G}$$

where

S = carried load (throughput)

G = offered load

The maximum (0.184) occurs when G = 0.5. Tanenbaum [Ref. 2: pp. 122–123] provides a complete derivation.

(2) Slotted ALOHA. The major revision to ALOHA is called slotted ALOHA. With this method, a station can only start to transmit at the beginning of a slot (frame time). Because there is the possibility of only one collision per frame time, this improves efficiency to about 36.8%, again excluding protocol overhead.

The equation describing transmission efficiency is similar to the one above. The maximum (0.368) occurs when G = 1.0.

$$S = Ge^{-G}$$

(3) CSMA. The next major improvement was Carrier Sense Multiple Access (CSMA). The concept was pretty simple, but revolutionary. In essence, the protocol said, "Listen to the medium before transmitting. Transmit if no other station is using the medium; wait if another station is transmitting." Although the change is relatively minor, the performance improvement over ALOHA is not. CSMA was further enhanced by adding collision detection and collision avoidance options.

(4) CSMA/CD. CSMA/CD (Carrier Sense Multiple Access with Collision Detection) took carrier sensing a step further. Since the host could listen before it transmitted, it could also listen while transmitting to see if the received signal is significantly different from the one it is transmitting. If it differs, that means there must have been a collision and the host stops transmitting the message. It then transmits a short error burst and initiates a binary exponential backoff algorithm which will determine how soon it attempts to sense the medium and begin to

transmit again. The other transmitting host executes the same algorithm. After the first collision, they each select a delay from the $2^1$ available. If they collide again, they each select from the $2^2$ available, and so on up to $2^{10}$. From that point on, the available number of delays remains the same and the stations try six more times. If they all fail, they stop trying and report a network failure to the data link layer. IEEE 802.3 is the CSMA/CD standard.

(5) *p*–persistent CSMA–CD. Although CSMA–CD represents a dramatic improvement over both incarnations of ALOHA, pure CSMA–CD is still far from ideal. In order to overcome some of the limitations in CSMA–CD a series of *p*–persistent CSMA–CD protocols were developed. The *p* denotes transmission probability. Common variants are non–persistent, 0.01–, 0.1–, 0.5– and 1–persistent. The basic protocol is that the sender listens to the medium and if it is busy, it waits. At this point, the various protocols diverge slightly.

Non–persistent CSMA–CD waits a random period of time before sensing the medium again. If the medium is clear, it sends its message. All the remaining variants continually sense the medium. The 0.01–, 0.1–, 0.5–persistent variants are slotted which means that there is a specified time increment and each will attempt to transmit in that time slot with probability *p*. In this discussion, $p$ = 0.01, 0.1 or 0.5. If the sender does not transmit in the first slot, it senses the medium and again attempts to transmit with probability *p*. The frame is transmitted after some delay. The 1–persistent variant constantly senses the medium so that it can transmit as soon as it is clear. It transmits with a probability = 1.

Channel efficiency is worst for 1–persistent CSMA–CD and improves as the value of *p* decreases. Nonpersistent CSMA–CD falls between the 0.1– and 0.01–persistent variants. IEEE 802.3 specifies 1–persistent CSMA–CD in spite of the poor channel efficiency (maximum of approximately 53%). The reason

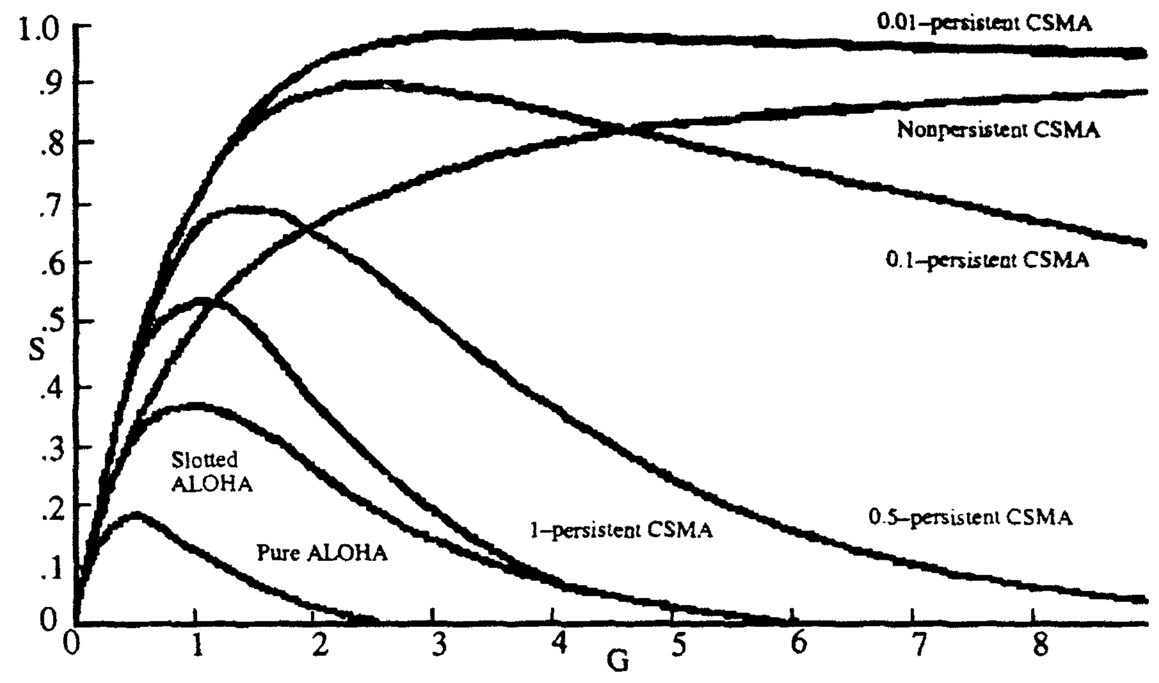


**Figure 2.3 Throughput of various contention protocols**

is that the transmission delay is inversely proportional to the transmission probability. While the 1-persistent CSMA-CD protocol will transmit immediately when it senses a clear channel, the 0.01-persistent CSMA-CD will wait an *average* of 100 time slots. A chart showing the throughput for various types of CSMA and ALOHA networks is shown in Figure 2.3 [Ref. 4: p. 304].

(6) CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance) takes a slightly different approach. Listening before transmitting is the same as with CSMA/CD, but the similarities end there. The collision avoidance scheme uses a dynamic "memory" to produce a seed for a random delay before transmission. In other words, the host waits for a specified minimum time and some additional pseudorandom time. If there is a large amount of traffic on the network and there are many collisions, the seed is larger, which causes the pseudorandom

delays to be larger. The converse is true for a lightly loaded network. CSMA/CA depends upon this pseudorandom delay to try to prevent collisions in the first place.

Because of the delays associated with collision processing, random transmissions and retransmissions, CSMA/CD and CSMA/CA are only suited for relatively short distances, typically less than 2.5 km.

### *b. Noncontention Protocols*

Noncontention protocols are based on the idea that transmissions could be more efficient if the physical layer did not have to use precious bandwidth to process collision information. While contention protocols are easier to implement and can be relatively efficient at low loads, they all reach a saturation point and eventually efficiency and throughput drop to zero.

The problems associated with wasted bandwidth and network grid lock led to several slotted implementations. Unfortunately, some of the slotted implementations are as wasteful as the protocols they were designed to replace. Early slotted protocols allowed a host to transmit for a specified period of time at a specified time. The drawback with these schemes was that if a particular host did not have any traffic to send, the bandwidth was wasted because no other host could use that time slot.

Eventually, two token passing schemes were developed. They were developed at approximately the same time in response to concerns about the ability to have a deterministic upper bound rather than one described by a random process. Primarily, these networks were to support manufacturing processes which required strict temporal control. One protocol was the token passing bus and the other was the token passing ring.

(1) Token Passing Bus. IEEE 802.4 describes the token passing bus. It was designed for installations which had a manufacturing process that used a bus

architecture. The token passing bus constructs a virtual ring. Each station has a table which indicates the next and previous station in the structure. Because the structure is in routing tables, hosts can join and leave the net at will and, eventually, the network will reconfigure itself [Ref. 2: p. 149].

This reconfiguration feature is implemented by attempting to pass the token to the next station. If the station does not answer, the sender generates another token and tries to pass it to the station it thinks is the successor. If the successor is active, it responds by seizing the token, otherwise, the originating station sends a solicit_successor message naming the station that was supposed to be the successor. The station after the successor sees that the successor is his predecessor, updates his table and responds. The originator updates his routing table and passes the token to the new successor [Ref. 2: p. 152].

New stations are added in a similar way. Occasionally, the token holder will ask if a host wants to join the network by sending a solicit_successor message. If a station answers, it becomes the token holder's successor. The original successor updates its routing tables and traffic begins anew [Ref. 2: p. 152].

Collision detection or avoidance schemes do not need to be implemented because a station can only transmit when it possesses the token. Since it is a bus structure, all stations get every message simultaneously (neglecting propagation delays).

(2) Token Passing Ring. In contrast to the stations on a token passing bus, stations are hardwired into position on the token passing ring and stations receive messages sequentially. No overhead is lost for ring maintenance, except in the case where the electronics unit fails, but the station is still powered.

Electrically, the token passing ring is much easier to set up because each station originates and terminates a point to point link. The transmitter for station 1 sends to station 2's receiver. Stations 2's transmitter sends to station 3's receiver and so on. To simplify wiring and troubleshooting, many token passing ring installations are wired in a star configuration with mechanical bypass relays in a central location (such as a wiring closet).

As defined in the 802.5 standard, the originating station is responsible for "draining" traffic off the ring. This means that the station that originates the message does not copy it through to the transmitter and send it again. Obviously we would not want the originating station to interfere with its own signal. For example, a 1 km ring operating at 4 Mbps data transfer rate (with propagation speed of 2 x $10^8$ m/s) will hold 20 bits if we neglect the one bit delay built into each station. Number of bits = ring length / propagation speed * data transfer rate. Since the token passing ring data packet is much larger than 20 bits, the originator is still putting data on to the ring when it starts to receive the beginning of its packet. After the originating station receives the last of its message, it sends the token on to the next station. This is known as Release After Reception (RAR). The data packet is not limited in size as are 802.3 (1500 bytes) and FDDI (4478 bytes), but there is a default limit of 5000 bytes based on a 10 ms token holding time and 4 Mbps data transfer rate.

In addition, a monitor station keeps track of ring performance as far as the originator removing traffic from the ring. When a packet passes the monitor station, it sets the monitor bit. If the originator drains the packet, the monitor does nothing. If the originator does not drain it for some reason, the monitor will.

(3) FDDI Standard. While the Ethernet standard is defined by IEEE, the FDDI standard is controlled by ANSI (American National Standards Institute).

The formal ANSI designation for FDDI is X3T9.5. As was the case for 802.3, the FDDI standard contains all the details necessary to implement the protocol including timing, framing and physical and electrical characteristics.

Currently, FDDI is a 100 Mbps medium, but standards committees within ANSI are working on an FFOL (FDDI Follow-on) which will push the standard to 400 Mbps and beyond. Even though FDDI was originally developed for optical fibers, the standard does not require its use. Several vendors support FDDI on copper wires.

FDDI uses a token passing technique similar to that of IEEE 802.5 to keep track of which station can transmit. The token grants a station the right to transmit and is 22 symbols (88 bits) long. It consists of the Preamble, Start Delimiter, Frame Control and End Delimiter which are described in detail under the Theoretical Timing Analysis section. The station holding the token can transmit until it has no more traffic, the token holding timer expires (asynchronous mode) or the station's synchronous bandwidth allotment is exhausted, whichever comes first. The actual protocol is more involved and will be explained in detail later.

FDDI describes the interactions of the physical layer, data link layer and network layer of the ISO OSI Seven Layer Model. The actual standards are broken down into the following documents: FDDI-PMD, FDDI-PHY, FDDI-PHY-2, FDDI-MAC, FDDI-MAC-2, FDDI-HRC, FDDI-LLC and FDDI-SMT. The FDDI-PMD, FDDI-PHY and FDDI-PHY-2 make up the physical layer, the FDDI-MAC, FDDI-MAC-2, FDDI-HRC and FDDI-LLC make up the data link layer and the FDDI-SMT straddles both the physical and data link layers. Figure 2.4 shows these relationships. Each will be described in detail below.

(a) FDDI-PMD. The FDDI-PMD document describes the Physical Layer Medium Dependent features of the FDDI standard. It describes the minimum transmitter power requirements and the minimum receiver sensitivity requirements as well as bit error probability. It also describes the physical interfaces between the medium (optical fibers or copper wire) and the electronics. Currently, two different PMD standards exist–one for multimode fibers (MMF) and another for single mode fibers (SMF). The most common implementation for FDDI is 62.5/125 micron multimode fiber (the optical fiber is 62.5 microns in diameter and the cladding is 125 microns in diameter). The fiber is diagramed in Figure 2.5. Multimode fiber is the preferred medium because it is less expensive to manufacture the fiber and the transceivers. Multimode fiber is also less technologically challenging, hence the lower cost. Multimode installations are limited to shorter link lengths because of the lower power output from the transmitters.The LEDs used in MMF transceivers emit light in a frequency band. The bandwidth and intensity vary, but must still satisfy the power budget, *i.e.*, if the LED doesn't put out enough power, the receiver may not be able to pick up the signal at the other end. The center frequency for 62.5/125 micron fiber is between 1300 and 1310 nm. The fact that the LED is polychromatic (simultaneously transmits light at different frequencies) makes the system a multimode variant. Multimode fiber reflects and refracts the light internally to guide the light to the receiving end. The losses associated with reflection and refraction coupled with the relatively lower power output from the LED limit the link length to 2 km or less (assuming that the bit error probability remains the same). The MMF–PMD addresses four areas of concern, namely: Optical power budgets using both active interfaces and bypassed interfaces, MIC (Media Interface Connector) mechanical mating requirements, multimode optical fiber requirements and the services provided to PHY and SMT.

Single mode fiber, on the other hand, uses a monochromatic laser LED as the light source. The laser LED has a higher output power and the optical

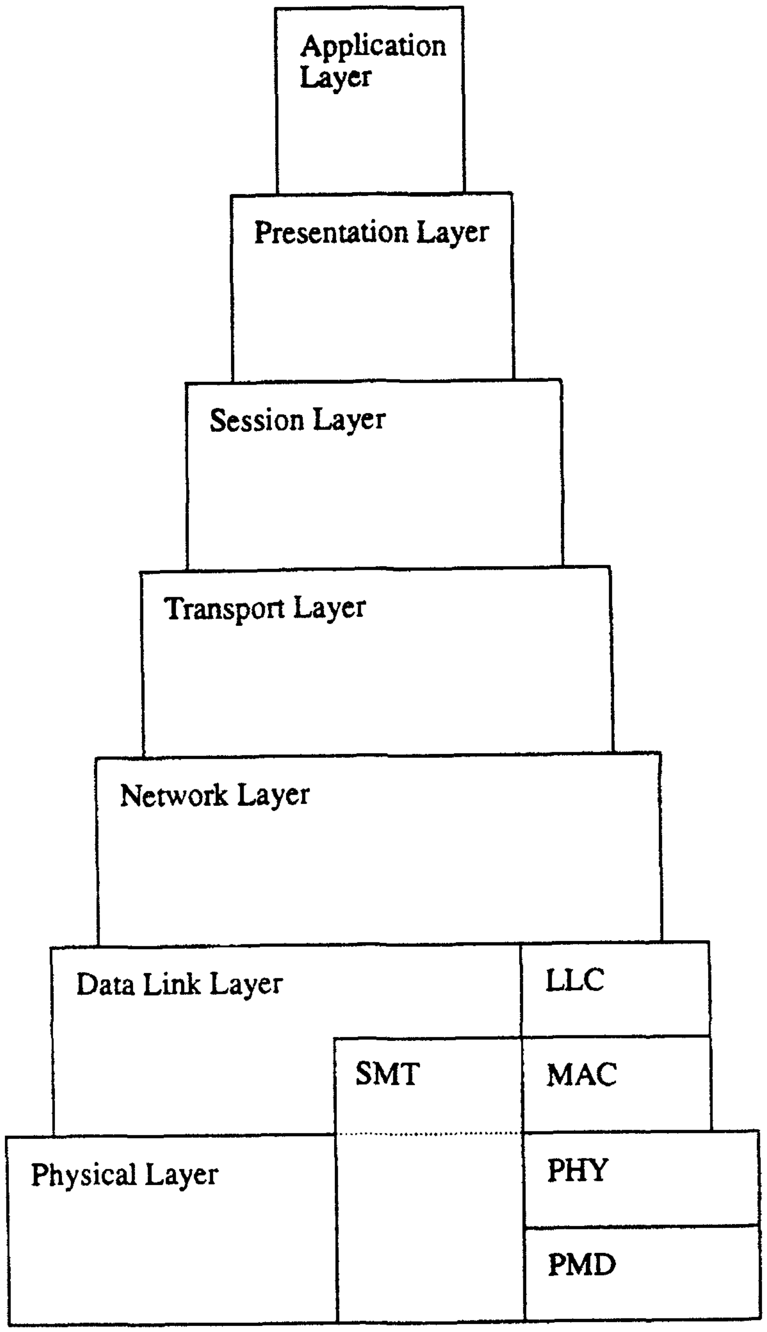


**Figure 2.4 FDDI functions within ISO OSI model**

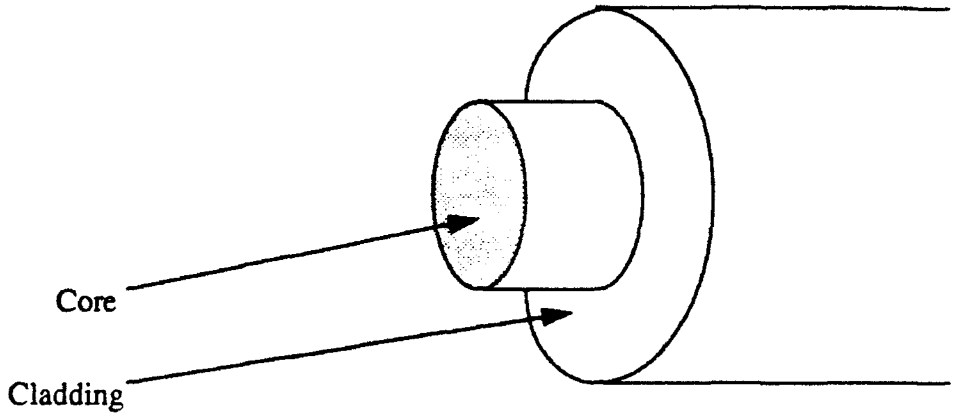


**Figure 2.5 Typical optical fiber construction**

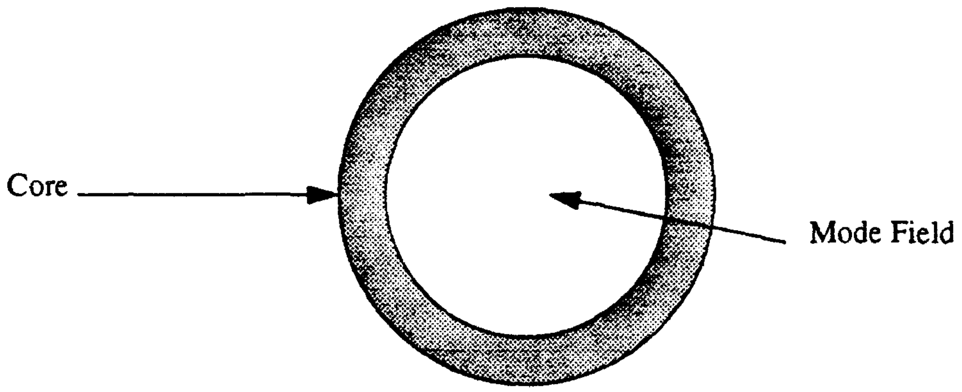


**Figure 2.6 Mode field for multimode fiber**

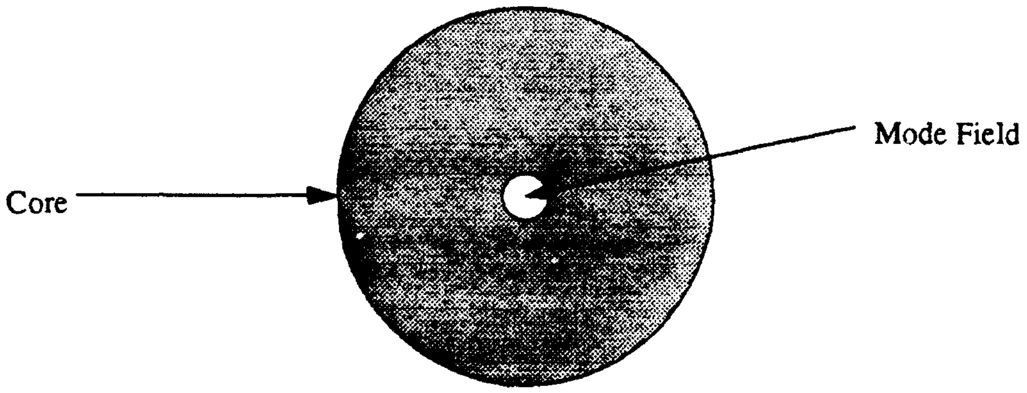


**Figure 2.7 Mode field for single mode fiber**

fiber has a much smaller mode field diameter. The single mode fiber mode field diameter is from 8.7 to 10.0 microns ± 0.5 microns [Ref. 5: p. 34]. By comparison, the mode field diameter of MMF is between 50 and 100 microns, depending on the fiber used. Figure 2.6 shows the multimode fiber mode field while Figure 2.7 shows the single mode fiber mode field. The extremely small mode field diameter ensures that only the fundamental wavelength is propagated down the fiber. This phenomenon is the same as radar ducting, in which atmospheric conditions trap SHF frequencies causing ship radars to detect other vessels at ranges in excess of 60 nautical miles. (The normal radar horizon is 20 to 25 nautical miles.) The higher output power and single propagation path combine to enable link lengths of up to 20 km.

(b) FDDI–PHY. The FDDI–PHY document describes the Token Ring Physical Layer Protocol. It deals with specifics concerning services provided to the MAC, services required of the PMD, services provided to the SMT, coding, transmitter and receiver operations, ring latency, line states, buffers, smoothers and filters.

One of the most interesting subjects covered in this document is the 4B/5B encoding scheme. 4B/5B encoding takes a hexadecimal symbol (four data bits) or a control symbol and encodes it with five transmission bits (on the physical medium). In reality, the data rate on the optical fiber is 125 Mbps, but the effective data transfer rate is 100 Mbps because of the 4B/5B encoding. Details of the 4B/5B code are provided in Appendix C. FDDI uses two layers of coding between the MAC and fiber. The first layer is the 4B/5B which takes a hexadecimal symbol and converts it to a code–group using the mapping shown in Appendix C. The format for this data is NRZ, which is Non Return to Zero, and means that a 1–bit is represented by a high voltage level and a 0–bit is represented by a low voltage level. The second layer of coding converts the NRZ format to a NRZI (Non Return to Zero Inverted). NRZI encoding is distinguished by a transition (either high to low or low to high) for each 1–bit and no transition for each 0–bit. Decoding occurs in the reverse order; the incoming code group is decoded from NRZI format to NRZ format and are then decoded from the NRZ format code group to hexadecimal symbols [Ref. 6: p. 21]. NRZI encoding directly supports the self clocking mechanism described next.

To successfully operate as a serial baseband system, FDDI is self clocking. The self clocking mechanism will only function properly if there are at least two transitions per symbol and no more than three consecutive zeros in each code–cell. The run length requirement also ensures that the maximum cumulative dc balance does not exceed ±10% [Ref. 6: p. 21]. Minimizing the dc balance makes interface components and circuit design easier.

In addition to extracting the clock signal from incoming data, encoding and decoding the symbols and buffering, PHY also takes care of detecting the line state and passing that information along to the MAC or SMT. PHY uses PM_SIGNAL.indication (a primitive from the PMD which indicates whether or not the received signal is above the detection threshold) as well as signals from Receive Function (Clock_Detect) and Elasticity buffer errors to determine the current line state [Ref. 6: p. 24]. PHY updates the line state when it changes. In the event that PHY can not immediately determine the line state, it reports Line–State_Unknown or Noise_Line–State.

(c) FDDI–PHY–2. The FDDI–PHY–2 [Ref. 7: pp. 1–2] document describes the Token Ring Physical Layer Protocol. It contains all of the information from the PHY document, but adds additional services to support HRC (Hybrid Ring Control) in addition to the standard MAC. PHY–2 can be used in place of PHY for standard MAC implementations, but PHY can not be used to support HRC. I will explain HRC more in later paragraphs.

(d) FDDI–MAC. The FDDI–MAC document describes the Token Ring Media Access Control protocol. The MAC is the lower sublayer of the ISO OSI Seven Layer Model data link layer. Specifically, the MAC takes care of tokens, framing, timers, priority messaging and error detection and recovery. I will discuss each of the major issues below. Some of the details follow in later sections.

The token gives a station the right to transmit. To get the token, a station must have traffic in either transmission queue. What happens next depends upon the specific implementation. If the ring is using synchronous scheduling, the station can send out its allocation of synchronous traffic. When the synchronous traffic is finished or the synchronous time allotment expires, the station may send asynchronous traffic provided that the token is early and the outgoing frame has a higher priority than the threshold. The station may transmit until the THT (Token Holding Timer) expires or there is no more asynchronous data in the queue, which ever comes first. For the case of a ring using only asynchronous scheduling, the token can be either restricted or nonrestricted. The nonrestricted token is considered the "normal" method. Time is dynamically allocated among all stations on the ring. A restricted token is used for an extended dialog between two stations. One example is an extended burst data transfer from a hugh speed device. Typically, this data transfer would last several times the TTRT (Target Token Rotation Time) [Ref. 8: p. 36].

Frame composition and size are discussed in depth in the Theoretical Timing Analysis section. The MAC is also responsible for maintaining timers to ensure that the ring functions properly. The most important timers are TRT (Token Rotation Timer), THT, TVX (Valid Transmission Timer) and TTRT. A little

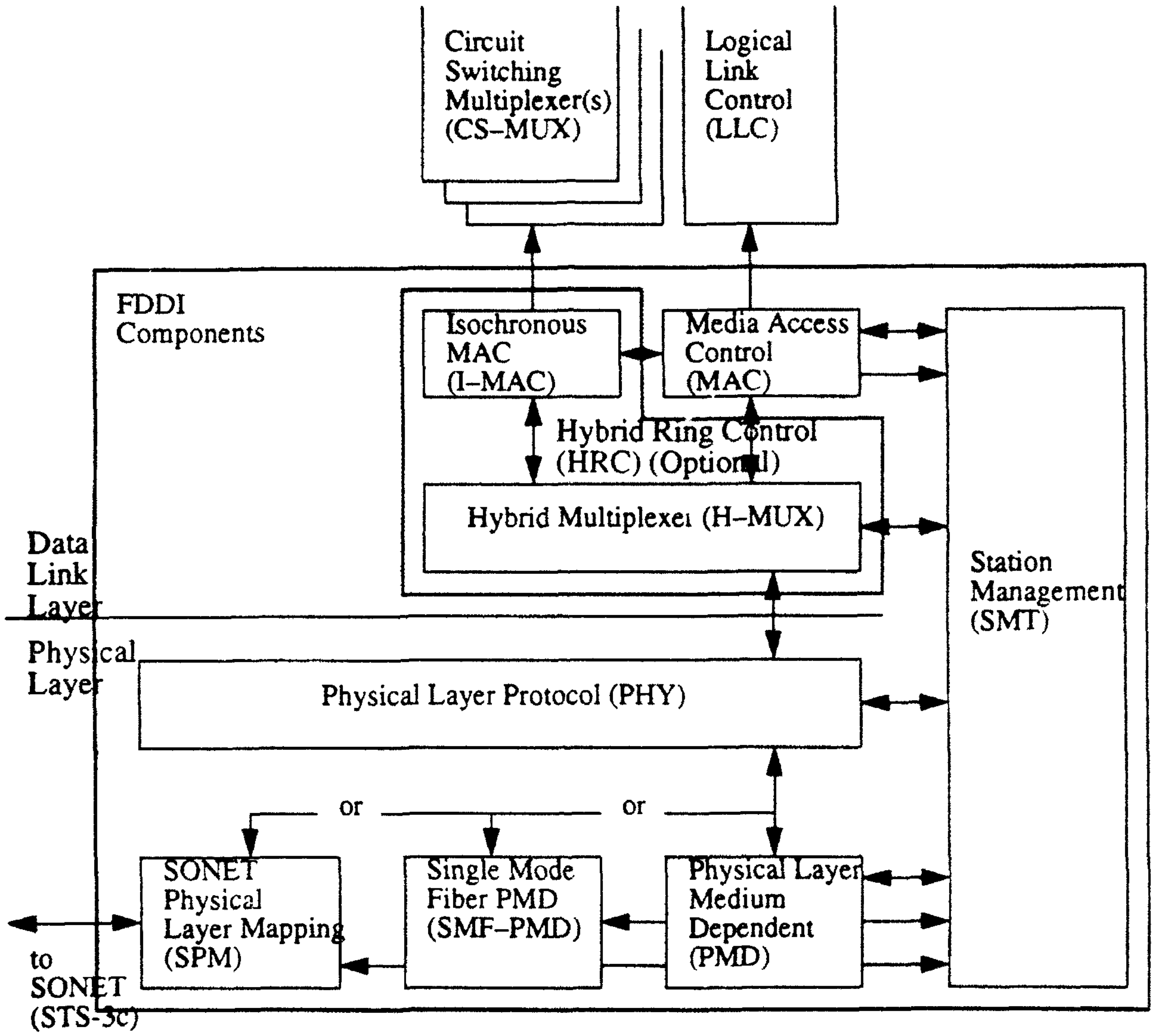

Circuit Switching Multiplexer(s) (CS–MUX)
Logical Link Control (LLC)
FDDI Components
Isochronous MAC (I–MAC)
Media Access Control (MAC)
Hybrid Ring Control (HRC) (Optional)
Hybrid Multiplexer (H–MUX)
Station Management (SMT)
Data Link Layer
Physical Layer
Physical Layer Protocol (PHY)
or
or
SONET Physical Layer Mapping (SPM)
Single Mode Fiber PMD (SMF–PMD)
Physical Layer Medium Dependent (PMD)
to SONET (STS-3c)

**Figure 2.8 FDDI Components Including HRC**

background will help explain the interactions between the timers. During ring initialization, each station sends out a frame with a requested token rotation time which is between that station's T_Min (default < 4 ms) and T_Max (default = 167.77216 ms). The MAC in each station looks at all of the requests and picks the lowest value. This value becomes T_Opr for the ring and once ring initialization is complete, T_Opr is copied into TTRT. TTRT is a constant during normal ring operation. This is the upper limit on how long it will take data to get from one station to another. TRT is used to time how long it takes for the token to reach the station again. When the station captures a token, it resets and restarts TRT. It then releases the token and times the rotation. If the timer rolls over, Late_Ct is incremented. TRT and Late_Ct determine whether or not the token is early, which determines if the station can send asynchronous traffic. THT determines how long a station may actually transmit asynchronous traffic. The mechanism works this way: if the token is early, the station copies TRT into THT, then copies TTRT into TRT and starts both timers running. The station actually gets the difference between TRT and TTRT to send asynchronous traffic. The station must stop transmitting when THT expires. It then releases the token. TVX allows transient ring error recovery. The default value is normally 2.62144 ms.

Priority messaging is accomplished by setting the three low order bits in the Frame Control field. Frame Control at the MAC layer is eight bits long and determines the type and class of service for the frame. The C bit (MSB) indicates the class of service, 0 for asynchronous and 1 for synchronous. The L bit indicates the address length, 0 for 16 bit addresses and 1 for 48 bit addresses. The ZZ bits together with the C and L bits determine the frame format. Table 1 provides the details (X indicates a Don't Care case, P indicates Priority and r indicates Reserved).

Table 1: FRAME CONTROL FORMAT BITS

| CLFF ZZZZ to ZZZZ | Frame type |
|---|---|
| 0X00 0000 | Void Frame |
| 1000 0000 | Nonrestricted Token |
| 1100 0000 | Restricted Token |
| 0X00 0001 to 1111 | Station management frame |
| 1X00 0001 to 1111 | MAC frame |
| XX01 r000 to r111 | LLC frame |
| 0X01 rPPP | LLC frame, asynchronous priority use |
| 1X01 rrrr | LLC frame, synchronous use |
| XX10 r000 to r111 | Reserved for implementer |
| XX11 rrrr | Reserved for future standardization |

The low order four bits are the hexadecimal control symbol and are used to describe different types of frames. The 0X01 rPPP series describes the priority system used for asynchronous frames. The lowest asynchronous priority is 000 and the highest asynchronous priority is 111 [Ref. 8: pp. 25–27]. Synchronous frames do not have a priority system because they are of equal priority and receive guaranteed service.

Error detection and recovery are accomplished using MAC Beacon Frames (1X00 0010) and MAC Claim Frames (1X00 0011). The MAC Beacon Frame tells stations on the ring that corrective action is required. The MAC Claim Frame is used during error recovery to determine which station gets to generate the token and send its traffic first [Ref. 8: p. 26].

(e) FDDI–MAC–2. The FDDI–MAC–2 document describes the Token Ring Media Access Control protocol. It contains all of the information from the MAC document, but adds additional services to support HRC (Hybrid Ring Control) in addition to the standard MAC [Ref. 9: pp. 1–2]. MAC–2 can be used in place of MAC for standard MAC implementations, but MAC can not be used to support HRC. Specifically, some of the enhancements add primitives which interface with the Hybrid Multiplexer (H–MUX) and associated primitives which support H–MUX operations like requesting specific classes of service.

(f) FDDI–HRC. FDDI–HRC describes the Hybrid Ring Control protocol. HRC was designed to support a mode of operation in which both packet switched and isochronous data are transmitted within the same frame structure. HRC consists of the Hybrid Multiplexer (H–MUX) and Isochronous Media Access Control (I–MAC). The H–MUX integrates the packet data and isochronous data into cycles which it transfers to PHY. I–MAC provides individual transmission channels for user isochronous data streams [Ref. 10: p. 1]. CAPSnet does not require this type of support and I will not provide any further details. HRC and its relationship to the rest of the FDDI Components is shown in Figure 2.8.

(g) FDDI–SMT. FDDI–SMT describes the FDDI Station Management functions. Specifically, SMT addresses the following areas: SMT services, SMT facilities, connection management and ring management. There must be exactly one SMT entity in each node, but there can be more than one PHY/PMD combination [Ref. 11: p. 13]. Two instances would be in the case of a DAS or a concentrator where there are two or more PHY/PMD combinations, but only one SMT that interfaces with all of them. The current version of SMT is 6.2 and is fully implemented in the LANplex 5012 concentrator.

Services. SMT provides services to PMD, PHY and MAC (it will also provide services to H–MUX when the standard is closer to completion) within a specific node. It also specifies services provided to System Management. The services that SMT provides take care of initializing the MAC protocol, controlling the MAC, gathering MAC status information, directing MAC to capture the next token, requesting line status from PHY, gathering PHY status information, providing control information to PHY, providing control information to PMD, directing PMD to join or leave the net and requesting optical signal strength from PMD [Ref. 11: pp. 21–30]. SMT services to System Management primarily consist of maintaining the Management Information Base (MIB).

Facilities. SMC facilities can best be described as a set of frames which support network management features as opposed to those services provided internally to a specific node. The first type of frame is the Neighborhood Information Frame (NIF) which SMT periodically sends to inform other SMT's of the sender's address and basic configuration. The Neighbor Notification protocol uses NIF's to find the logical upstream neighbor address (UNA) and, optionally, its logical downstream neighbor address (DNA) [Ref. 11: p. 129]. The next group of frames SMT sends are Status Information Frames (SIF) which are used to request and provide responses to requests from other SMT's regarding a station's configuration and operating information. The SIF Configuration request and SIF Operation request can be a unicast, multicast or broadcast [Ref. 11: p. 131]. The next group of SMT frames is an ECHO Frame (ECF) which is used for SMT–to–SMT loopback testing [Ref. 11: p. 134]. The next type of SMT frame is a Resource Allocation Frame (RAF) which is used to allocate resources, specifically synchronous bandwidth. Future expansion capability is provided for follow–on network services [Ref. 11: p. 135]. The next type of SMT frame is a Request Denied Frame (RDF) which is sent when any of several unsupported options or frame types have been requested from another SMT. The next type of SMT frame is an Extended Service Frame (ESF) and is used for extending and exercising new types of SMT frames [Ref. 11: p. 136]. The next type of SMT frame is a Status Report Frame (SRF) which a station uses to periodically announce Station Status which could be used by FDDI ring management software [Ref. 11: p. 137]. The last group of SMT frames is Parameter Management Frames (PMF) and it allows FDDI ring management software to actually alter entries in the MIB [Ref. 11: p. 138]. Some SIF's, some ECF's, the ESF and all PMF's are optional.

Connection Management. Connection management (CMT) consists of managing resources associated with PHY. CMT performs the following functions [Ref. 11: p. 151]:

- Establishing and initializing physical connections
    - Invoking a Path Test
    - Controlling the Optical Bypass Switch
    - Connection Continuity Test
    - Withholding undesirable or illegal connections
    - Signalling physical topology information
    - Providing local loop configuration with neighbor MAC
- Station configuration control
    - Placement of available MAC entities
    - Support for configuration policies
- Detecting faults at the physical layer
    - Continuous link error monitoring
    - Reconfiguration around physical layer faults
- Support for logical fault tracing function
- Removing orphan MAC PDU's (Protocol Data Units)

Ring Management. Ring Management (RMT) performs a similar suite of services for the MAC entities that CMT performs for the PHY entities. RMT performs the following functions [Ref. 11: p. 151]:

- Detecting faults at the MAC layer
    - Identifying stuck beacons
    - Detecting duplicate addresses
    - Resolving duplicate addresses that prevent Ring_Op
- Initiating the logical fault tracing function
- Notifying MAC availability for data services

## D. MULTICASTING AND DISTRIBUTED COMPUTING

Multicasting is a means of sending the same message to several recipients without having to send the same message repeatedly. A broadcast message is a special kind of multicast. At first, the distinction may sound trivial because each station on the network "listens" to all the data. In a macro sense, this is true, but in reality, the station only "listens" long enough to determine whether or not that particular data stream is for it.

### 1. Issues

Multicasting can occur at any of the upper five layers of the ISO-OSI Seven Layer model. The data link layer performs the address resolution function. Many distributed applications can use multicasting to more efficiently use available bandwidth. Algorithms are available for resource (or process) location, fault tolerance, redundant information storage and message delivery (in dynamic multicast groups).

#### *a. Implementation*

Each host has a unique numeric hardware address that identifies it as a valid entity on the network. Both FDDI and Ethernet use a 16– or 48–bit hardware address. The 32–bit IP address is uniquely bound to the hardware address. Each host also has a common or alpha name, or the complete domain name for a TCP/IP network. There is a one–to–one correspondence between the alpha name and the numeric address. If one host wants to send information to another, it must unambiguously identify the recipient.

Within each host, there may be one or more applications that need to access the network. These applications can operate in layers four through seven of the OSI Seven Layer Model. We will assume that all applications that need to access the link operate from the Application Layer. From the human point of view, this assumption makes sense because this is where the human interfaces with the machine. If we were to consider a real-time tactical data system, this assumption might not be valid because a sensor could be feeding raw data to layer six or five.

(1) Bit Structure. Each frame needs to know where to go when the source releases it. In a general way, broadcasting and unicasting are simple implementations because they only use one address in each case. For FDDI or Ethernet using 48–bit addressing, the broadcast address is FFFFFFFFFFFF$_h$. All the unicast addresses are in the range 000000000001$_h$ to 7FFFFFFFFFFF$_h$ (remember that 000000000000$_h$ is not a valid machine address). This leaves the range of multicast addresses as 800000000000$_h$ to FFFFFFFFFFFE$_h$. This equates to approximately 1.4072 x $10^{15}$ addresses for unicast and 1.4072 x $10^{15}$ addresses for multicast. FDDI further restricts this range by half because bit 46 (bit 0 is the LSB and bit 47 is the MSB) indicates whether an address is locally or universally assigned. If 16–bit addressing is used, only 32767 addresses are available for unicast and 32767 are available for multicast. (There is no similar universal versus local distinction when using a 16–bit addressing scheme.) Using 16–bit addresses does not present a problem for a small subnet like the CAPSNet, but in a large distributed system, it could be limiting

Even though we've discussed the mechanics or constraints in general terms, we have not really talked about the way multicasting is really implemented. In the next section, we discuss how hosts send messages to multicast addresses.

(2) Multicast Tables. In order for multicasting to work correctly, we must have some way to identify where a particular process resides. If we cannot locate the process, it will be impossible to implement multicasting. Usually, the data

link layer holds the multicast table which cross references a process with a particular multicast address. Ahamad presents an algorithm which performs this resource finding. His performance analysis (cost in CPU time) showed considerable improvement when using multicasting instead of broadcasting for large numbers of multicast addresses (few processes per address) [Ref. 27: p. 199].

The most common way of implementing a multicast table is to have a list of addresses at each destination. This reduces traffic on the net. Even with FDDI, efficiency is an important consideration; it is always expensive to waste bandwidth [Ref. 27: p. 193]. Since we know that the destination address field of each message gets to all the hosts on the net, it is reasonable for each one to have a "guard list" of those addresses they are supposed to copy. Each host knows which messages it is supposed to get and it only needs to look for specific addresses in addition to its broadcast and unicast addresses [Ref. 27: p. 195].

In order for this to work, there must be some mechanism which sets up the multicast table in the first place. To provide maximum flexibility, the multicast table must also be dynamic. Spanning tree algorithms have been used in routing messages to static multicast groups. This method, Wall's algorithm, is an inefficient method in a dynamic environment. Belkier and Ahamad proposed an algorithm for incrementally updating subgraphs in the tree rather than reconstructing it entirely. Incremental updating provides a cost saving when a high percentage of the members change in the multicast groups [Ref. 28: p. 110].

(3) Example. Let's say that Host A has processes 1, 2, 3, and 4 running, Host B has processes 1, 3, 5 and 6 running, Host C has processes 2, 3, 4 and 6 running and Host D has processes 2, 4 and 6 running. Further, we will assume that Host A is the primary host for process 1. Host A needs help with process 1 because it and Host B are getting data faster than they can process it. Host A will send a

message to Host D telling him to add 9000000000A$_h$ (the multicast address for process 1) to his multicast table. Host D adds this address to his multicast table and tells the data link layer to make sure it gets messages with that address. Host D tells Host A that the multicast address is in his table and is ready to accept data. The next time Host D sees $9000000000A$_h$ in the destination address, he reads the message and processes the data. In the same vein, when an object associated with a certain multicast address migrates, the original processor tells its data link layer to reject the address and the new processor tells its data link layer to accept the address [Ref. 29: p. 425].

This method works well, but there are limitations. Each host has a limited amount of memory. This means that the multicast tables are limited in size [Ref. 27: p. 194]. If the multicast table is limited to 10 entries, what happens when process 1 is the 11th process? The host can check on all active processes and delete those that have not had any activity over the past *n* seconds, or it can send the multicast to another host which will "forward" the data to the appropriate host. Of course, the other hosts may experience the same problem.

Several methods exist for process identification, namely: broadcasting, using mailboxes, forwarding agents, message routers and other hybrid schemes. The comparison of round trip packet times for these techniques versus multicasting is provided in Table 2 [Ref. 29: p. 429].

**Table 2: ROUND TRIP COMPARISON**

| Technique | Round trip time | (seconds) | |
|---|---|---|---|
| | No migration | 1 Migration | 2 Migrations |
| Mailbox | 11.89 | 11.84 | 11.86 |
| Broadcast | 6.75 | 6.74 | 6.73 |
| Forwarding | 6.71 | 8.77 | 11.16 |

Table 2: ROUND TRIP COMPARISON (Continued)

| Technique | Round trip time | (seconds) | |
|---|---|---|---|
| | No migration | 1 Migration | 2 Migrations |
| Inform clients | 6.78 | 6.76 | 6.78 |
| Message router | 11.78 | 10.89 | 11.77 |
| Delayed informing | 6.73 | 6.73 | 6.73 |
| Name server | 6.75 | 6.79 | 6.79 |
| Multicast | 6.71 | 6.73 | 6.74 |

## E. THEORETICAL TIMING ANALYSIS

As discussed in section II.B.2, Ethernet data transfers represent a random process with a hard lower bound, rather than a deterministic process. FDDI data transfers represent a random process as well, but transfers in this case have a hard upper and lower bound. When analyzing an FDDI net, the maximum transfer time can be computed *a priori*, provided that the network specifics (number of nodes, length of links, etc.) are known. The same is not true for Ethernet as the experimental data in Chapter V seems to suggest.

The next several sections will cover the 802.3 and FDDI protocols, and provide the basis for calculating the ideal (theoretical) values for data transfers for each protocol. In section V.B, these values will be compared to the experimental results.

### 1. IEEE Standard 802.3

Ethernet is one implementation of IEEE standard 802.3. The standard describes all of the aspects of the network including physical connections, tolerances, primitives that support higher layers, electrical specifications and a host of other details.

### a. *Physical Characteristics*

IEEE standard 802.3 specifies a transmission bandwidth of 10 MHz which corresponds to 10 megabits per second (Mbps) data transmission rate. When data is placed on the physical medium, it leaves at 10 Mbps. Because of the bandwidth requirements, special cabling must be used for all but the shortest links. Three types of media are available: thick, thin and twisted pair. Thick and thin cables refer to coaxial (called 'coax') cables that have specified characteristics.

Thick coax, also called 10Base2, is usually yellow (as the standard recommends) and is marked every 2.5 meters for transceiver taps. The distance between transceiver taps is specified to reduce interference (reflections) and improve performance. The following equations derive the distance between taps for Ethernet. The frequency (or bandwidth) is $f$ (in Hertz), $c$ is the propagation speed in a guided media (in meters per second) and $\lambda$ is the wavelength (in meters).

$$f = 10 \cdot 10^6$$

$$c = 2 \cdot 10^8$$

$$\lambda = \frac{c}{f}$$

$$\lambda = 20$$

From the equations above, it is clear that one wavelength is 20 meters. Because of physical properties of transmission lines, best performance is realized when taps are located at binary fractional intervals. One eighth of 20 is 2.5 meters. Therefore, taps are at 1/8$\lambda$, 1/4$\lambda$, 3/8$\lambda$, 1/2$\lambda$, etc.

Unshielded twisted pair, similar to modular telephone cable and called 10BaseT or UTP, is the least expensive of the alternatives, but is limited to short runs in areas with relatively low levels of background RF (radio frequency) radiation. The limitations are due to the gauge of the wire (usually 24 or 26 ga.) and the fact that it is unshielded. The signal level also tends to be lower than the coaxial versions because of the lower bandwidth available.

### *b. Framing*

As with other protocols, 802.3 has specific frame parameters. The next several paragraphs will cover the various parts of the 802.3 frame. These details are important when we calculate the theoretical data transfer times. The frame itself is divided into several parts which are: preamble, start delimiter, destination address, source address, length of data field, data, pad and checksum.

(1) Preamble. The preamble is a minimum of seven bytes. The sequence is $AA_h$ ($10101010_b$). This allows the sender and receiver to synchronize their clocks. Clock synchronization is important because the transmission delays are random. The other important reason for synchronization is that 802.3 uses Manchester encoding which specifies data–bearing transitions in the middle of bit times.

(2) Start Delimiter. The start of frame delimiter is a one byte field that tells the receiver that the frame is beginning. The sequence is $AB_h$ ($10101011_b$).

(3) Destination Address. The destination address field is either two or six bytes long. For CAPSnet, the destination address is a full six bytes.

(4) Source Address. The source address field is two or six bytes. As for the destination address, it is also six bytes long.

(5) Length of Data Field. The length of data field indicates how many bytes of data follow. Valid entries are in the range 0 to 1500, inclusive. Because of propagation delays and sensing requirements, the minimum data field length is 46 bytes. Any deficiencies in the data field are corrected in the pad field.

(6) Data Field. As explained above, the data field can be from 0 to 1500 bytes long.

(7) Pad Field. The pad field is only used when the length of the data field is less than 46 bytes. Consequently, the pad field can be from 0 to 46 bytes long. If the data field exceeds 46 bytes, this field is ignored.

(8) Checksum. The final field is the checksum. It is a 32–bit CRC (Cyclic Redundancy Check) code that detects and corrects all burst errors of length 16 or less, all errors with an odd number of bits and all single and double errors. It can also detect 99.997% of all 17–bit burst errors and 99.998% of all 18–bit and longer error bursts. It uses a generator polynomial of degree 16 and is called CRC–16. The polynomial is $x^{16} + x^{15} + x^2 + 1$.

### c. *Timing and Overhead Calculations*

Now that we have covered the overhead associated with 802.3, we can compute the overhead associated with each 802.3 frame. As we shall see, in some cases, the overhead is excessive. Each frame must have the fields described above. Total overhead for each frame is 7 bytes (preamble) + 1 byte (start delimiter) + 6 bytes (destination address) + 6 bytes (source address) + 2 bytes (data field length) + 4 bytes (checksum) = 26 bytes. Other 802.3 implementations that use 2–byte vice 6–byte address fields would have an 18 byte overhead.

Based on everything to this point, we can determine how long 802.3 should take to transmit a message of arbitrary length. From our discussion on TCP/IP, we know that they will use the MTU associated with the physical transmission system in use. The MTU for 802.3 is 1500 bytes. Backing up through the protocol stack, we find that the size of the IP header is 20 bytes, which leaves 1480 bytes for data from the TCP layer. TCP also adds a 20 byte header leaving 1460 bytes of data for each TCP packet. This becomes our jumping off point. Assume that only the sender has a message to send, there are no transmission or noise losses, propagation delays are negligible, the receiver does not acknowledge any of the frames until it receives the last one and that the message is 58080 bytes. Our goal is to find out how long it will take to transmit this message and how much overhead is included.

Since the message starts at the TCP layer, we need to divide the total message size (58080 bytes) by the effective packet size (1460 bytes) which gives us 39 full packets with 1180 bytes left over. The 1500 byte packet that IP gives to 802.3 fits in the data field and the 26 header bytes must be included. When the packet actually moves on the cables, it is 1526 bytes long. The time to transmit one packet is 1.221 ms (1526 bytes x 8 bits/byte ÷ 10 Mb/s), which means that the time to transmit all 39 full packets is 47.611 ms. Similarly, it takes 0.965 ms to send the last packet. We add the two figures to get 48.576 ms to transmit the message. Now we must include the delay between subsequent transmissions while the sender listens to the link. The minimum delay is 51.2 µs between each packet, of which there are 39. The total delay is 1.997 ms which gives us 50.573 ms for the entire transfer. Similar calculations are used to find the theoretical transfer times for the large and small files. Table 3 summarizes the results for large, medium and small file transfers.

**Table 3: ETHERNET THEORETICAL TRANSFER TIMES WITHOUT ACKS**

| File size | Time (s) |
|---|---|
| Large (1155959 bytes) | 1.007084 |
| Medium (58080 bytes) | 0.050573 |
| Small (11 bytes) | 0.000062 |

A more realistic figure would include the time it takes for the receiver to send back the ACKs for the packets received. To find out how much additional time that will take, we will further assume that the ACK will fit in a 72 byte 802.3 packet and that ACKs are sent after every fourth incoming packet. Each ACK takes

0.1088 ms to transmit (including the 51.2 μs delay) and there are 10 ACKs for a total of 1.088 ms. The total transmission time is now 51.661 ms. As above, similar calculations are used to find the theoretical transfer times for large and small files with ACKs. Table 4 summarizes the results for all three file transfers.

**Table 4: ETHERNET THEORETICAL TRANSFER TIMES WITH ACKS**

| File size | Time (s) |
|---|---|
| Large (1155959 bytes) | 1.028626 |
| Medium (58080 bytes) | 0.051661 |
| Small (11 bytes) | 0.000170 |

Calculating the number of overhead bytes is relatively straightforward. We will continue to use the 58080 byte file for our calculations. From our discussion above, we know that there are 39 1526–byte blocks and one 1206–byte block. This gives us a total of 60720 bytes. Overhead is everything that is not part of the original message. In this case it is 2640 bytes or 4.55%. Again, we follow a similar process with the large and small file. Table 5 summarizes the results.

**Table 5: ETHERNET OVERHEAD BY BYTES AND PERCENT**

| File size | Bytes | Percent |
|---|---|---|
| Large (1155959 bytes) | 52272 | 4.521960 |
| Medium (58080 bytes) | 2640 | 4.545455 |
| Small (11 bytes) | 66 | 600 |

Notice the excessive amount of overhead associated with the small file transfer. This is a direct result of the size of the TCP, IP and Ethernet headers. With TCP and IP "tuned" to the Ethernet packet size, the least percentage of overhead achievable is 4.52055%. Transferring any file larger than 1460 bytes will result in at most 9.035% overhead. Now consider the case of a protocol stack based on the full ISO OSI Seven Layer Model. We can safely assume that the Ethernet portion will remain the same, but instead of two layers which will add 20 bytes of overhead, there are now five! The result is nearly *1200 percent* overhead. Obviously, this is an extreme case because many of the applications that people use (*e.g.* E–mail or file transfers) move relatively larger chunks of data around. One application that is adversely affected by large overhead is a distributed, real–time system.

One final area for consideration is throughput. Throughput is best defined as the number of user data bits transferred per second. Calculating throughput is simple (we will continue with the previous example). We divide the message size (in bits) by the calculated delivery time. We sent 464640 bits in 51.661 ms (time calculated with ACKs) which gives our system a throughput of 8.994 Mbps. Table 6 summarizes the theoretical throughput values for the large, medium and small file sizes.

**Table 6: ETHERNET THEORETICAL THROUGHPUT**

| File size | Mbps |
|---|---|
| Large (1155959 bytes) | 8.990312 |
| Medium (58080 bytes) | 8.994054 |
| Small (11 bytes) | 0.516432 |

Unfortunately, as we know from our previous discussion on 1–persistent CSMA–CD, Ethernet's maximum throughput is only slightly better than 5 Mbps.

## 2. FDDI Standard

The next few sections describe various aspects of FDDI including the physical characteristics of the optical fibers and drivers, topology, framing and timing characteristics.

### *a. Physical Characteristics*

The most common FDDI implementation is on 62.5/125 micron optical fibers, as discussed in the FDDI–PMD section.

(1) Topology. FDDI is implemented as a ring topology. Actual installations may look like a star because of other factors, such as concentrators and physical constraints such as wiring closets. Concentrators will be discussed in the next section. The FDDI standard defines two distinct options for implementation: a single ring structure and a dual ring structure. Usually, the dual ring structure is installed in areas where increased bandwidth or network availability is crucial. The single ring structure is installed in other cases.

(a) Single ring structure. The single ring structure supports 100 Mbps data transfer rates between a host (a SPARCstation 2 in the case of CAPS) and a concentrator. Traffic flows in only one direction on the optical fiber. A host connected to a single ring is called a Single Attach Station (SAS). The port on the concentrator is called a master while the one on the remote host is called a slave. Each port is configured with a duplex receptacle which connects the entities to the optical fiber. One side of the receptacle is the transmitter and the other is the receiver. Polarity is important because the ring topology must be maintained; the plugs and receptacles are polarized for this reason. Polarized plugs and receptacles also provide an added benefit–the network is easier to install and modify. I will discuss topology and the ring vs. star layout in the concentrator section.

(b) Dual ring structure. The dual ring structure is composed of two counter–rotating rings. In locations where availability is critical, both rings carry the same data, but in different directions. One ring is designated as the primary (whose ports are labelled A) and the other is the secondary (whose ports are labelled B). The plugs and receptacles are polarized the same way as the ones for the single ring structure. Hosts on the dual ring are called Dual Attach Stations (DAS). If either ring fails, network integrity is maintained because the hosts switch to the alternate ring. In the case of a hardware failure, the hosts "upstream" and "downstream" stop sending traffic to the failed station and electronically connect the primary and secondary rings. This condition is known as a "wrap" because the ring wraps along itself. This combination of dual counter–rotating rings and wrapping provides a robust implementation for critical networks. In cases where the two rings are carrying different data (higher throughput), the rings will still wrap, but the data rate will be cut in half. The network will still continue to function, though.

(2) Concentrators. A concentrator takes the dual ring structure and converts it to a single ring structure. DAS's require two ports on the concentrator, one for each of the A and B connections, while SAS's only require one port. The following example should be helpful to get a feel for how this works. The Port 1 transmitter is associated with the A ring and sends traffic to Server 1. The server 1 transmitter passes the traffic to the Port 1 receiver. Port 1 passes the traffic to Port 2. The Port 2 transmitter sends traffic to Server 2. The Server 2 transmitter sends traffic to the Port 2 receiver. Port 2 sends its traffic to Port 3 and so on. Eventually, Port *n* will send its traffic to Port 1 and the whole process starts over. Notice that station *n* is connected to Port *n* (with a duplex optical fiber). This is why the network will look like a star even though it is a ring.

In addition to providing a convenient building block for fiber optic networks, many concentrators support more than one type of physical layer protocol. For example, concentrators from Synemetics and Cabletron allow switching between FDDI and Ethernet networks. Two types of concentrators are available: smart and dumb.

(a) Smart Concentrators. Smart concentrators include a microprocessor which allows it to perform network management tasks as well as providing an easy expansion path for either DAS's or SAS's. These models are more expensive because of their added functionality. Smart concentrators allow for hot-swapping modules and automatic reconfiguration for failed stations.

(b) Dumb concentrators provide a common connection point for SAS's and allow easy expansion. These are commonly called hubs.

(3) Cable Lengths. The maximum length of an uninterrupted multimode fiber is 2 km (approximately. 1.2 miles). This ensures that the Probability of Bit Error ($P_e$) is less than $2.5 \times 10^{-10}$. Even though the optical fiber is an excellent transmission medium, the LEDs have a maximum output power and there is some loss due to boundary layer diffusion and multimode effects. Single mode optical fibers used with lasers extend the single link limit to 20 km (approximately 12 miles). Again, the limit is imposed to ensure that the worst-case $P_e$ remains less than $2.5 \times 10^{-10}$.

(4) Token Passing. FDDI uses a system similar to the one used by IEEE 802.5 (token passing ring) to designate which host on the network gets to transmit. Exactly one token exists on the ring at one time and the station that possesses the token may transmit. If synchronous traffic is not supported, the host which has the token may transmit until the token holding timer (THT) expires provided that the token is early and the outgoing frame priority is higher than the threshold. This is similar to IEEE 802.5, but the remaining ring characteristics which follow are quite different. The THT varies for each station and between successive token receipts for any arbitrary station, depending upon the instantaneous value of TRT. The station must pass the token on after it can no longer transmit.

If synchronous traffic is supported, the station can transmit its allotment of synchronous traffic before enabling THT. Synchronous traffic provides guaranteed bandwidth. If the token was early, that is if TRT is less than TTRT and Late_Ct is zero, and the priority for the queued frame is higher than the T_Pri threshold, the station may then transmit asynchronous traffic until the THT expires. The station must then pass the token to the next one on the ring. If the token was late, the station must pass the token on after sending its allocation of synchronous traffic.

FDDI specifies a Release After Transmission (RAT) method of passing the token. Again, this characteristic differs from the 802.5 ring which uses the Release After Reception (RAR) method. This ensures that the maximum amount of bandwidth is available for data and was instituted because of the larger ring sizes involved with FDDI.

(5) Each station on the FDDI ring introduces a 15 symbol (60 bit) delay into the ring. This allows each station to check the first three fields of each frame to determine if it is a valid token. If so, and the station has frames to send, the station removes it from the ring (by not propagating it) and sends the frames.

### *b. Framing*

In many respects, the FDDI frame definition is like the 802.3 frame definition. In order for the network to provide useful services, many of the same components are required. The following paragraphs describe each of the fields in the FDDI frame [Ref. 8: p. 25].

(1) Preamble. The preamble is a minimum of 16 symbols. The sequence is $11111_b$. It is transmitted as five bits because FDDI uses 5B/4B encoding to improve reliability and DC balance. The preamble allows the receiver to synchronize its clock with the incoming data stream. Clock synchronization is

required to reduce the probability of losing data. The other important reason for synchronization is that 802.3 uses Manchester encoding which specifies data-bearing transitions in the middle of bit times.

(2) Start Delimiter. The start of frame delimiter is a two symbol field that tells the receiver that the frame is beginning. The sequence is JK ($1100010001_b$).

(3) Frame Control. The frame control field provides important control information. Specifically, it describes the frame class, address length and format. The frame class bit indicates either a synchronous or asynchronous frame. The address length bit indicates either a 16– or 48–bit address field. The format bits, in conjunction with the frame class and address length bits describe the frame type. The frame types available are:

- Void frame
- Nonrestricted Token
- Restricted Token
- Station management frame
- MAC frame
- LLC frame
- Reserved frame types

(4) Destination Address. The destination address field is either four or twelve symbols long. For CAPSnet, the destination address is a full 12 symbols. All stations must be capable of recognizing and acting on 16–bit (four symbols) addresses. Stations which operate on a 16–bit addressing scheme must also be capable of recognizing 48–bit address frames and correctly reacting to the following conditions:

- Repeating 48–bit address frames
- Recognizing 48–bit broadcast address
- Claim Frames with 48–bit addresses
- Beacon Frames with 48–bit addresses

Stations which use the 48-bit addressing scheme must have a minimum 16-bit address capability which allows:

- Fully functional 16-bit address
- Recognizing 16-bit broadcast address

(5) Source Address. The source address field is either four or twelve symbols long. As with the destination address, it is also 12 symbols long.

(6) Data Field. The data field can contain 0 or more symbol pairs, *i.e.* an even number of symbols. Since the whole FDDI frame is limited to 9000 symbols, the data field can not be larger than 8956 symbols (4478 bytes).

(7) FCS. The next field is the Frame Check Sequence or checksum. It is a 32-bit CRC (Cyclic Redundancy Check) code that detects and corrects all burst errors of length 16 or less, all errors with an odd number of bits and all single and double errors. It can also detect 99.997% of all 17-bit burst errors and 99.998% of all 18-bit and longer error bursts. It uses a generator polynomial of degree 32 and is called CRC-32. The polynomial is $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^{8} + x^{7} + x^{5} + x^{4} + x^{2} + x + 1$.

(8) Ending Delimiter. The ending delimiter is a two symbol field consisting of two T symbols ($01101_b$) for the Token and only one T symbol for the data frame.

(9) Frame Status. The frame status field is a minimum of three symbols long and conveys information to follow-on stations or the originating station. The three indicators in the frame status field are:

- Error detected
- Address recognized
- Frame copied

Their functions are relatively self-explanatory.

### c. *Timing and Overhead Calculations*

Now that we have covered the overhead associated with FDDI, we can compute the overhead associated with each FDDI frame. As was the case with 802.3, in some cases, the overhead is excessive. Each frame must have the fields described above. Total overhead for each frame is 8 bytes (preamble) + 1 byte (start delimiter) + 1 byte (frame control) + 6 bytes (destination address) + 6 bytes (source address) + 4 bytes (checksum) + 0.5 bytes (end delimiter) + 1.5 bytes (frame status) = 28 bytes. FDDI implementations that use 2–byte vice 6–byte address fields would have a 20 byte overhead.

Based on everything to this point, we can determine how long FDDI should take to transmit a message of arbitrary length. From our discussion on TCP/IP, we know that they will use the MTU associated with the physical transmission system in use. The MTU for FDDI is 4478 bytes. Backing up through the protocol stack, we find that the size of the IP header is 20 bytes, which leaves 4458 bytes for data from the TCP layer. TCP also adds a 20 byte header leaving 4438 bytes of data for each TCP packet. This becomes our starting point. Assume that only the sender has a message to send, there are no noise losses, propagation delays are negligible, THT is 5 ms, and that the message is 58080 bytes. Our goal is to find out how long it will take to transmit this message and how much overhead is included.

Since the message starts at the TCP layer, we need to divide the total message size (58080 bytes) by the effective packet size (4438 bytes) which gives us 13 full packets with 454 bytes left over. The 4478 byte packet that IP gives to FDDI fits in the data field to which 28 header bytes are appended. When the packet actually moves on the fiber, it is 4506 bytes long. The time to transmit one packet is 0.3605 ms (4506 bytes x 8 bits/byte ÷ 100 Mb/s), which means that the time to transmit all 13 full packets is 4.6865 ms. Similarly, it takes 36.32 μs to send the last packet. We add the two figures to get 4.72 ms to transmit the message. Since the entire message can be sent without THT expiring, there are no token rotation delays. In the case of a larger file transfer, or a shorter THT, we would have to take the token circulation time into account. Similar calculations are used to find the theoretical transfer times for the large and small files. Table 7 summarizes the results for large, medium and small file transfers.

**Table 7: FDDI THEORETICAL TRANSFER TIMES WITHOUT ACKS**

| File size | Time (s) |
|---|---|
| Large (1155959 bytes) | 0.093740 |
| Medium (58080 bytes) | 0.004722 |
| Small (11 bytes) | 0.000062 |

A more realistic figure would include the time it takes for TCP at the receiver to send back the ACKs for the packets received. To find out how much additional time that will take, we will further assume that the ACK will fit in a 68 byte FDDI packet and that ACKs are sent after every fourth incoming packet. Each ACK takes 9.04 μs to transmit (including the 3.6 μs station delays) and there are four ACKs for a total of 36.16 μs. Now we must add the 0.00088ms for each token of which there are seven. The total transmission time is now 4.764 ms. As above, similar calculations are used to find the theoretical transfer times for large and small files with ACKs. Table 8 summarizes the results for all three file transfers.

**Table 8: FDDI THEORETICAL TRANSFER TIMES WITH ACKS**

| File size | Time (s) |
|---|---|
| Large (1155959 bytes) | 0.094318 |
| Medium (58080 bytes) | 0.004764 |
| Small (11 bytes) | 0.000063 |

Calculating the number of overhead bytes is relatively straightforward. We will continue to use the 58080 byte file for our calculations. From our discussion above, we know that there are 13 4506–byte blocks and one 454–byte block. This gives us a total of 59032 bytes. Overhead is everything that is not part of the original message. In this case it is 952 bytes or 1.64%. Again, we follow a similar process with the large and small files. Table 9 summarizes the results.

**Table 9: FDDI OVERHEAD BY BYTES AND PERCENT**

| File size | Bytes | Percent |
|---|---|---|
| Large (1155959 bytes) | 17748 | 1.53534857 |
| Medium (58080 bytes) | 952 | 1.63911846 |
| Small (11 bytes) | 68 | 618.181818 |

Notice the excessive amount of overhead associated with the small file transfer. This is a direct result of the size of the TCP, IP and FDDI headers. With TCP and IP "tuned" to the FDDI packet size, the least percentage of overhead achievable is 1.532%. Transferring any file larger than 4438 bytes will result in at most 3.06375% overhead. Now consider the case of a protocol stack based on the full ISO OSI Seven Layer Model. We can safely assume that the FDDI portion will remain the same, but instead of two layers which will add 20 bytes of overhead, there are now five! The result is nearly *1200 percent* overhead. Obviously, this is an extreme case because many of the applications that are commonly used (*e.g.*, E–mail or file transfers) move relatively larger chunks of data around. One application that is adversely affected by large overhead is a distributed, real–time system.

One final area for consideration is throughput. Throughput is best defined as the number of user data bits transferred per second. Calculating throughput is simple (we will continue with the previous example). We divide the message size (in bits) by the calculated delivery time. We sent 464640 bits in 51.661

ms (time calculated with ACKs) which gives our system a throughput of 8.994 Mbps. Table 10 summarizes the theoretical throughput values for the large, medium and small file sizes.

**Table 10: FDDI THEORETICAL THROUGHPUT**

| File size | Mbps |
|---|---|
| Large (1155959 bytes) | 98.047796 |
| Medium (58080 bytes) | 97.531486 |
| Small (11 bytes) | 13.924051 |

Luckily, we do not see the same problems with FDDI that we see with Ethernet as far as throughput is concerned. Rather than continue to discuss these theoretical results in a vacuum, the next section is devoted to comparing the two sets.

## 3. FDDI vs. Ethernet Theoretical Results

In general, we should expect to see a one–order–of–magnitude performance increase when we compare FDDI and Ethernet. Because of differences in the protocols, this may not always be the case. I will compare file transfer times, both with and without ACKs, overhead, and theoretical throughput, and provide comments where appropriate.

### *a. Comparison of File Transfer Times (without ACKs)*

The results in Table 11 show what we expect to see from a theoretical point of view: FDDI transfers are between 8.7 and 9.7 times faster. When viewed strictly from the physical layer point of view, this is correct because FDDI transmits data at 10 times the rate of Ethernet. One big drawback with only looking at transfers from this perspective is that it does not include any processing time required by the layers above the Physical Layer in the protocol stack. This comparison also does not include ACKs, which higher layers will need to ensure that the file transfer is completed properly.

**Table 11: COMPARISON OF ETHERNET AND FDDI THEORETICAL TRANSFER TIMES WITHOUT ACKS**

| | Time (s) | Time (s) | Percentage Improvement |
|---|---|---|---|
| File size | Ethernet | FDDI | |
| Large (115595 bytes) | 1.007084 | 0.093740 | 974.33 |
| Medium (58080 bytes) | 0.0505728 | 0.004722 | 971.00 |
| Small (11 bytes) | 0.0000616 | 0.00000632 | 874.68 |

We can improve the model slightly by adding acknowledgments into the picture. The next section discusses that comparison in detail.

### b. *Comparison of File Transfer Times (with ACKs)*

Table 12 shows the comparison between FDDI and Ethernet with ACKs. There is a modest improvement for the large and medium size files because the ACK takes less than 10% of the time on FDDI than it does on Ethernet. This is a result of the smaller packet size used on FDDI and the shorter time between token arrival and ACK departure. The FDDI ACK packet is six bytes shorter than the Ethernet ACK packet because of the 46 byte minimum data field size.

**Table 12: COMPARISON OF ETHERNET AND FDDI THEORETICAL TRANSFER TIMES WITH ACKS**

| | Time (s) | Time (s) | Percentage Improvement |
|---|---|---|---|
| File size | Ethernet | FDDI | |
| Large (115595 bytes) | 1.0286264 | 0.094318 | 990.60 |
| Medium (58080 bytes) | 0.0516608 | 0.004764 | 984.40 |

Table 12: COMPARISON OF ETHERNET AND FDDI THEORETICAL TRANSFER TIMES WITH ACKS (Continued)

| | Time (s) | Time (s) | Percentage Improvement |
|---|---|---|---|
| File size | Ethernet | FDDI | |
| Small (11 bytes) | 0.0001704 | 0.0000114 | 1394.74 |

This is about as far as we can take the theoretical analysis without resorting to a probabilistic description. File transfers can not take place any faster than what has been shown here. This model does not address the situation where more than one station is ready to send traffic at a time. It also does not address collisions and the subsequent resolution process for Ethernet. That said, I will now move on to an overhead comparison.

### c. *Comparison of Overhead*

Table 13 shows that for larger file transfers, FDDI offers less overhead because of the larger data field in the packet. For data transfers of less than 1460 bytes, FDDI actually incurs a penalty because of the frame status field at the end of the packet which allows physical layer ACKs. The figures shown agree with common sense because the FDDI data field is almost three times larger than the Ethernet data field.

Table 13: COMPARISION OF FDDI AND ETHERNET OVERHEAD

| | Ethernet | FDDI | Percentage Improvement |
|---|---|---|---|
| File size | Bytes | Bytes | |
| Large (115595 bytes) | 52272 | 17748 | 194.52 |
| Medium (58080 bytes) | 2640 | 952 | 177.31 |
| Small (11 bytes) | 66 | 68 | -2.94 |

The final topic of interest is a comparison of throughput, which is discussed next.

### *d. Comparison of Throughput*

The final area of interest is that of throughput. As with the transmission times, we should expect an improvement of approximately 10 times. Table 14 shows that for the large and medium size files, that there is almost that much improvement. As with the transmission times, the Ethernet throughput does not take collisions and collision processing into account. What this illustrates is that theoretically, FDDI represents a 10–fold performance increase. Under normal network operating conditions, as the EPA warns, "Actual mileage may vary."

**Table 14: COMPARISON OF THEORETICAL THROUGHPUT FOR ETHERNET AND FDDI**

| | Ethernet | FDDI | Percentage Improvement |
|---|---|---|---|
| File size | Mbps | Mbps | |
| Large (115595 bytes) | 8.99031174 | 98.0477958 | 990.59 |
| Medium (58080 bytes) | 8.99405352 | 97.5314861 | 984.40 |
| Small (11 bytes) | 0.51643192 | 13.9240506 | 2596.20 |

Conducting this analysis is not a total waste of effort because it is a starting point for trying to determine where the protocols break down and how much time they add to a "normal" file transfer. To be able to determine some of the other unknowns in the equation, we will next look at the equipment and discuss how quickly or slowly each piece interacts with the others. In Chapter V, I will again pick up the performance thread and compare the actual results with those presented in this chapter.

# III. EQUIPMENT

The CAPS project hardware consists of two Sun servers and five Sun SPARCStation 2's. As discussed earlier, this equipment and the Synemetics LANplex 5012 concentrator form an FDDI network. In the following sections, we will look at the hardware aspects of this project in detail.

All of the computers in the CAPS lab are SPARCs which means that they are RISC (Reduced Instruction Set Computers) based systems. RISC technology enables the processor to work faster because it has fewer choices and the internal circuitry is therefore simpler. The processors are all CMOS (Complementary Metal Oxide Semiconductor) construction which uses less electrical power and generates less heat. The servers and workstations operate at a 40 MHz clock speed.

## A. SERVERS

The servers are different multiprocessor models from the same Sun line. I will cover the larger one first and the smaller one second. In this case, larger refers to the physical size rather than processor power.

Both servers are built around a dual S-Bus and VME bus architecture. This architecture provides extra capabilities as far as expansion is concerned. It also allows us the flexibility of using vendors other than Sun to support further upgrades to the equipment.

### 1. SPARCserver 690 MP

The SPARCserver 690 MP is a four processor server which can support 640 megabytes of main memory and 52 gigabytes of IPI disk storage space. The 690 used to support CAPS has 128 megabytes of memory and approximately 8.9

gigabytes of disk storage. The SPARCServer 690 MP is called suns7 and is operating in the single processor mode under SunOS 4.1.2 in order to maintain software compatibility with ONTOS 2.2. ONTOS 2.2 is the design database engine for CAPS and is not yet compatible with multiprocessor machines.

### 2. SPARCserver 630 MP

The SPARCserver 690 MP is a four processor server which can support 128 megabytes of main memory and 26 gigabytes of SCSI disk storage space. The 690 used to support CAPS has 80 megabytes of memory and approximately 5.2 gigabytes of disk storage. The SPARCServer 630 MP is called suns5 and is operating in the multiprocessor mode under SunOS 4.1.2M. Suns5 is also the YP master for the CAPSNet.

## B. WORKSTATIONS

All five workstations are Sun SPARCstation 2 models. The processor is a Sun proprietary RISC design, as are those of the SPARCServers. Each SPARCstation has 64 megabytes of main memory (expandable up to 128 MB) and between 1.2 and 1.9 gigabytes of SCSI disk storage space. All of the workstations have color monitors. They all operate on SunOS 4.1.1 which is a superset of BSD Unix 4.

## C. CONCENTRATOR

As described earlier, the concentrator is an intelligent hub which supports connections for both Single– and Dual–Attach Stations. In addition to taking care of physical connections, it also provides protocol conversion, intelligent management and numerous other administrative capabilities [Ref. 38: p. 3]. A complete functional description will be provided below. M. Coden et. al. provide a good overview of hub concentration [Ref. 21: pp. 22–39].

### 1. Functional description

Functionally, the LANplex 5012 performs the following tasks: hub concentration, attachment conversion, maintenance features and management. Each of these functions will be described in the paragraphs below.

#### *a. Hub Concentration*

Hub concentration brings all of the network connections to a single point rather than spreading them out along a backbone. It allows the network administrator to more easily perform his duties by having all the network connections in one place rather than distributed throughout a floor or an entire building.

Hub concentration also has a more profound impact on FDDI; it changes the physical network topology from a ring to a star. The FDDI network maintains the logical ring structure. The advantages to this type of system are that network maintenance and installation are greatly simplified. Adding or removing a host is as easy as plugging or unplugging a duplex optical fiber. Other advantages will be discussed in subsequent sections. A disadvantage is that fiber links tend to be longer than necessary because they all go to a common connection point. For example, adjacent workstations (1 to 2 m apart) are not connected with a 3 or 5 m fiber, but two 20 m fibers. This increases the propagation delay between eight and 13 times.

#### *b. Attachment Conversion*

Attachment conversion refers to the ability of the concentrator to accept connections from both Single– and Dual–Attach Stations and handle each properly. This feature is important because SAS interfaces cost about $2000 each while DAS interfaces cost about $10,000 each. Pricing aside, a DAS interface is more complicated than a SAS interface and a DAS requires twice as many fibers per station.

The concentrator allows much more flexibility in this regard. Both DAS's and SAS's can be connected to the concentrator provided that they are not connected in an illegal configuration. In general, valid connections are: A to B, B to A, S to S, S to M, M to A, M to B, M to S and M to M, where A is an A port, B is a B port, M is a master port and S is a slave port [Ref. 11: p. 18]. This can be easily controlled by the management capabilities built into the concentrator.

### c. *Maintenance Features*

Maintenance features refer to the ability to repair the concentrator or modules without completely disabling the network. Modularity is the main feature which permits this type of functionality. Power supplies, processors, ports and interface units are all modular and can be duplicated or replaced easily. The ability to "hot–swap" modules is discussed under Backplane Intelligence in the following section.

Modules are easily replaced by unlocking the retaining handles and pushing outward. The fan module is also easily replaced or repaired.

### d. *Management*

Management refers to the ability to control the network once it is operational. Management features include designating which port attaches to a particular network structure, port loading, error detection and analysis and fault isolation.

The management system allows the processor to detect new stations added to the ring or old stations which have been removed. In the same general scheme, the management system designates which MAC is currently in use on which data path.

## 2. Capabilities

After evaluating the capabilities of several different systems, we decided that a concentrator needed to perform several functions and have some basic capabilities. We broke these requirements into four different areas of conc•rn: system design, backplane intelligence, data paths and data throughput. We will discuss each of these areas in the following paragraphs.

### *a. System Design*

The following items were considered essential to our needs for an FDDI concentrator: third generation modular system, scalable bandwidth, integrated management and fault tolerance. Specifically, we wanted FDDI modules which would support both DAS's and SAS's, optional ethernet multiplexing capability, campus FDDI connectivity, dual MAC structure, robust system processor and a standby system processor.

Ethernet connectivity was an issue because of the installed Ethernet base. We also did not want to give up our ability to interface with the rest of the campus or outside world via our electronic links. Because of FDDI's bandwidth, we did not want to waste bandwidth on overhead. Ethernet connectivity had to be "intelligent." Rather than encapsulating an Ethernet package inside an FDDI packet, we wanted the Ethernet packets to be converted to FDDI packets. This type of implementation is better for our installation where many users remotely login to the subnet to work.

The system processor module is particularly important because it provides the foundation for the concentrator's functionality. In addition to providing services to support integrated management and fault isolation (discussed below), it is also needed to support a remote update capability, control system configuration, and provide at least two out-of-band management channels. The remote update capability permits uploading new software to RAM to check system operation. If there are no problems, the RAM can be written to flash EPROMs for future use. In this way, updates can be provided electronically (via FTP or other binary file transfer) by the vendor, rather than waiting for conventional methods. The out-of-band management capability allows the network administrator to remotely login to the concentrator from an Ethernet, RS-232 or other serial line connection to correct problems or update the configuration. These are functions that would normally be performed remotely in our installation.

Each of the general system requirements will be discussed in detail in the following paragraphs.

(1) Third Generation System. The third generation system is one which includes network management functions, a dedicated processor for operating the concentrator, and switching and translation technology. It also refers to the scalability and expandability available to the end user. By comparison, first generation systems were passive or active devices that only provided concentration for one type of physical layer implementation. These devices did not include a processor. They were simple hubs. Second generation systems included a processor to add network management capabilities to the hub. Network management includes some kind of port management in the form of diagnostics and routing. This means that individual ports can be attached to specific data paths, within the constraints of the concentrator. For example, if the concentrator has two internal data paths, ports A, B, D, F and G could be attached to data path 1, while ports C, E and H could be attached to data path 2. If data path 2 failed for some reason, the concentrator could be quickly reconfigured to add ports C, E and H to data path 1. Third generation systems, take the second generation capabilities and expand them by providing additional management capabilities, additional physical layer protocols, switching

technology and translation. The biggest changes revolve around the additional physical layer protocols and translation, and switching technology. Adding more physical layer protocols means that the concentrator can be used as an intelligent bridge. Adding translation capabilities on top of that means that packets coming from one physical layer and going to another are transmitted in the native protocol rather than encapsulated in the destination protocol format. This distinction directly impacts network performance. For example, consider many Ethernet packets going to a host attached to the FDDI subnet. The data portion of the Ethernet packet is 1500 bytes while that of the FDDI packet is 4478 bytes. Nearly three Ethernet packets will fit in one FDDI packet, which in turn reduces the overhead by nearly three times. Switching technology incorporates high-speed switching for those cases where data must be transferred internally between modules or data paths.

(2) Scalable Bandwidth. Scalable bandwidth refers to the ability to change the scope or scale of the network fairly easily. Three basic areas concerned us: increasing the number of hosts attached to the FDDI concentrator, allowing several independent networks and the ability to support FDDI data transfer rates in excess of 100 Mbps. We wanted an FDDI concentrator that would support more than the five SAS's and two DAS's (seven or nine FDDI ports, depending upon configuration) that are currently connected to the subnet when the time came to increase its scope. We expected host migration to the FDDI subnet, in much the same way that NPS has migrated from mainframes and large minicomputers to powerful workstations. Simplicity was also a factor that we considered when we looked at adding hosts. It would be unrealistic to expect to have to buy a new chassis to add more stations, within reason, of course. An ideal concentrator would allow interconnections between the four most common types of networks available. The minimum requirement for us was to be able to support FDDI and Ethernet on

independent internal data paths. The concentrators on the market do that to varying degrees. The job of increasing network throughput had to be relatively easy. In other words, we did not want to reengineer or replace the concentrator's backplane to handle data rates higher than 100 Mbps when the FDDI standard changed. The ability to support higher bandwidth is designed into the system from the ground up and is not a feature that can be added later.

(3) Integrated Management. Integrated management is the ability to provide network management as a built–in function rather than an afterthought. For example, the ability to isolate a module or a port for testing should be relatively unobtrusive, except for the hosts attached to that module or port, and should be relatively easy to perform. It is also important to be able to isolate a single host for testing and/or network administration. We did not want to take the subnet down every time we added or removed a host from the subnet. The integrated management function also ties in very closely with the fault tolerance requirement discussed in the next paragraph.

(4) Fault Tolerance. Fault tolerance is important from the aspect of network availability. The concentrator must be able to provide fault isolation at the very least. In this scenario, the concentrator removes the offending host from the subnet by "wrapping" the ring. A more capable concentrator removes the offending host from the ring by transferring it to another connection for testing and informs the network administrator that there is a problem with that host. The concentrator also must be able to diagnose internal failures (data path failures, module or functional failures within modules and primary processor failures). As with the integrated management capabilities, these features must be relatively easy to set up and use, and must provide for "graceful" degradation.

### *b. Bcckplane Intelligence*

The requirements for backplane intelligence were: "hot-swap" capability, module slot independence, identical diagnostic and configuration control for each slot and supporting network addressing.

(1) "Hot-swap" Capability. The ability to "hot-swap" modules is important for providing maximum availability. This means that the concentrator does not have to be off to insert and remove modules. If one of the interface modules failed, we wanted to be able to replace it without turning the concentrator off and disabling the entire subnet. Hot swapping is supported by several ASIC's (Application Specific Integrated Circuits) which detect which slots are active and which are not. When a module is removed, the ASIC's connect the data path on the backplane rather than allowing it to go through the module. When the [illegible]e is replaced, the ASIC's reroute the data so that the data path goes through the module again.

(2) Module Slot Independence. This simply means the ability to put any module in any slot and have it work properly. As the network grows in size and complexity, we did not want to be burdened with moving boards around in the concentrator to make sure each was "in its proper place."

(3) Identical Diagnostic and Configuration Control. Each slot must have the same diagnostic and configuration capabilities. This ties in very closely with the ability to put any module in any slot. The system would not meet our needs very well if only two slots provided FDDI diagnostic support (our installation requires a minimum of three FDDI modules) and three slots provided Ethernet diagnostic support, etc.

(4) Supporting Network Addressing. The ability to support network addressing is important for providing in band or out of band management for the concentrator. Although the concentrator can be operated from the front panel, operations from a console are much easier. In band management refers to the ability to telnet to the concentrator from a station attached to the concentrator (FDDI in this case) and perform management functions. Out of band management refers to the abiiity to use a local console or a workstations from a network other than the one under the concentrator's control.

### *c. Data Paths*

As a minimum, we wanted to have three FDDI data paths in the concentrator. This type of configuration would allow us to have an internal dual ring architecture with an additional ring that could be used for local testing or as a backup for a failed data path.

Ethernet connectivity was also required for backwards compatibility. Again, three data paths were specified to permit a one-to-one mapping with the FDDI data paths. In this way, we could dedicate one Ethernet data path for each FDDI data path to support rapid reconfiguration.

Support for other networks was a plus, but not a requirement.

### *d. Data Throughput*

The backplane had to support an aggregate data transfer rate of 800 Mbps. Essentially, this meant that the manufacturer had to build-in extra capacity to support higher data rates than v'hat is currently specified. Even though the current FDDI standard only supports a 100 Mbps data rate, future improvements to FDDJ will push that to 400 Mbps and beyond.

### 3. Purchase considerations

Several important issues were discussed prior to deciding which company we would use to supply the equipment. The issues we addressed were: compatibility with Sun equipment, compatibility with FDDI standards, company size, company performance (*i.e.*, track record), and deliverability. I will discuss each of these issues in detail below.

#### *a. Compatibility with Sun Equipment*

Our initial concern was with compatibility. A. Kahn of AMD discusses interoperability issues between different vendors using the AMD FDDI chipset, but does not specifically discuss the vendors [Ref. 19: p. 266]. We had purchased the VME cards for the SPARCServer 690 and SPARCServer 630 and were in the process of purchasing the S–bus FDDI cards for the SPARCStations. At the time, there were no other vendors which could deliver an S–bus FDDI card for the SPARCStations. With those basic decisions made, we moved on to other considerations. To ensure compatibility, we contacted the OEM's that Sun had used [Ref. 24: p. 5] to conduct their tests. Sun used the following OEM's to test the S–bus FDDI cards:

- AT&T
- DEC
- Timeplex
- Synoptics
- Sumitomo
- Synernetics

This narrowed our initial search effort and allowed us to choose an OEM that we knew, in advance, had completed a successful compatibility test. Sun was not in a position to offer a recommendation on which concentrator they preferred. I contacted each of the companies listed above and received detailed product information from DEC, Timeplex, Synoptics and Synemetics.

### *b. Compatibility with FDDI Standards*

We were not interested in purchasing a concentrator which did not implement FDDI properly or in accordance with the published standards. We also wanted to make sure that the concentrator supported TCP/IP, NFS and all other services that our Ethernet currently supports.

TimePlex, one of the vendors we originally contacted, removed their equipment from the market before the close of the bidding process. They were unable to have their equipment function reliably.

### *c. Company Size*

Company size was an issue because we did not want to purchase a piece of equipment, have the company go out of business and have an orphan on our hands. On the other hand, we did not want to purchase equipment from a company that would not be responsive to our requests or queries. We were also not interested in a company that was so small that we would not receive adequate support because of a lack of manpower or resources. Of all the companies contacted, Synemetics was the only small one. All the others were much larger in terms of operating capital, name recognition and sales.

### *d. Company Performance*

Company performance was important because of concerns about product availability and compatibility. The OEM's track record was considered essential in deciding which company we thought would give us the best value. Several companies had working versions of their hardware operating at InterOp in the summer of 1991. Another issue was software support and software production.

#### e. *Deliverability*

Finally, availability was considered. The OEM that we selected had to have a sufficient quantity of concentrators on hand to allow 30 day delivery. The concentrators must be in production. Even though Sun may have been able to use a prototype for their compatibility tests, we wanted to have a production model so that as many bugs had been worked out as possible. Synernetics was producing one of the few concentrators on the market and the only third generation concentrator.

## D. FDDI INTERFACES

Having an FDDI concentrator without the proper interfaces would have left us in an impossible situation. Shortly after I began researching FDDI concentrators, Sun announced their S bus cards for the SPARCStations. The interface cards provide the conversion between the light signals on the optical fibers and the electrical signals the computer needs to operate properly. The interface card also provides clock signal retrieval, signal strength indications and appropriate buffering.

### 1. Sun FDDI/DX Interface Card

The Sun FDDI/DX interface card provides dual FDDI attachments (A and B ports) for the SPARCServer 600 series servers and other VME bus based products. The interface card is built around Motorola's 68020 32-bit microprocessor and interfaces with the VME bus. The 68020 is responsible for managing the card, providing supervisory functions, controlling Direct Virtual Memory Access (DVMA) and providing real-time network data processing.

The interface card includes 256 kilobytes of dedicated RAM for processing incoming and outgoing FDDI packets and an additional 256 kilobytes of RAM dedicated to input and output buffers for the FDDI transceivers. These functional areas are connected by local system and data buses to reduce loading on the VME bus.

### 2. Sun FDDI/S Interface Card

The Sun FDDI/S interface card provides a single FDDI attachment (S port) for the SPARCStation 1+ and 2 series workstations. It will also work in the SPARCServer. The FDDI/S card connects to the SBus using a custom ASIC chip. DMA transfers are also supported. Each FDDI/S interface card has its own MAC to allow it to function without a concentrator when connecting two SPARCStations.

## E. JUSTIFICATIONS FOR CHOICES

Our first choice for a concentrator system was one supplied by Synernetics. Our second choice was Cabletron. Several factors were overriding in our final decision. The most important factor for us was to have a piece of equipment that was tested. As mentioned earlier, the equipment we bought needed to work with the Sun computers we had in the lab. Basically, those were the six vendors listed earlier (AT&T, DEC, Timeplex, Synoptics, Synernetics and Sumitomo). We knew that each of these vendors had some product that was interoperable with the interface cards that we were going to get.

In order to pare the field further, I contacted each of the companies and asked for information on their FDDI products. AT&T did not respond and I was never able to find out what their product was like in great detail. Sumitomo also provided insufficient information to be able to support a purchase decision.

DEC provided quite a bit of technical information. Their concentrators did not meet our needs for several reasons. First, their concentrators were only first generation devices (a hub). Because it was only a hub, it did not support the management features we felt were important for effective network operation. Second, they did not provide an easy way to bridge FDDI to Ethernet. In order to perform that function, an additional chassis was required. Third, their hub was limited to eight FDDI ports. Again, if we wanted to expand in the future, we would be required to buy another chassis and the modules to populate it. Finally, DEC did not support the "hot swap" capability.

Timeplex offered a third generation concentrator (TimeLAN 100 Concentrator*32) with integrated management facilities and redundant processors. It also supported up to 32 SAS's, but it was not in full production when we needed to order. It, like DEC's concentrator, did not have a built-in bridge from FDDI to Ethernet. I have since discovered that Timeplex has removed their concentrator from the market because of difficulties associated with their SMT implementation.

SynOptics offers the LattisNet 2914 which supports up to 14 SAS's. This concentrator is a second generation device which is not modular. It does provide the intelligent management features that we wanted. Like DEC's and Timeplex's concentrators, it does not provide any bridging capability. They were one of the vendors who demonstrated their product at InterOp on 1990.

Essentially, the company that met all the technical requirements that we agreed were important was Synernetics. In addition to having proven hardware on the market for over a year, they are primarily responsible for writing the SMT software that provides the important services for the FDDI ring. They license this software to virtually every other FDDI concentrator vendor.

At the beginning of the section, I mentioned Cabletron. At the time we made the purchase decision, Cabletron was in the process of putting the final touches on their FDDI concentration cards. They were still in a Beta test status and had not been used in a commercial environment. Their reputation as a networking leader deserved our consideration. The major factor against them was that their equipment had only been prototyped and not actually used in other than a laboratory environment.

## F. PURCHASE PROCESS

Once we decided which features were important for our installation, we wrote up the appropriate documentation for a sole-source procurement. To make a long process short, the sole-source procurement was disapproved and we had to send it out for bids. While we were in the sole-source loop, Synemetics had licensed its LANplex backplane technology to 3Com. At the end of the bidding period, we had five bids: one from Synemetics, three from 3Com vendors and one from RAYCOM. In essence, we received four bids for the same equipment and one from an unknown source.

We rejected the bid from RAYCOM because their delivery date was the end of September, which was not going to be within 30 days of the contract award date. From the information we received, it appeared that their product was only an FDDI to Ethernet bridge. We accepted the bids from the other vendors because the equipment was the same. In the end, we bought the concentrator from Synemetics.

# IV. INSTALLATION

The overall philosophy of the changeover from Ethernet to FDDI was to allow the CAPSNet to continue to function while we installed the hardware. We followed this principle to the greatest degree possible during the change. To that end, we planned the installation in three phases. Phase I was a simple connection between the two servers. In this configuration, there was no "control" on the FDDI ring. In essence, the only function that this satisfied was ensuring that the driver software and the FDDI interface cards could communicate. This phase was completed by 17 July 1992. Phase II was a dual ring connection between the servers using the Synemetics LANplex 5012 concentrator. This was the first phase in which we had a fully managed and functional FDDI network, even though the servers were the only resources connected to the FDDI ring. This phase was completed on 23 July 1992. Phase III was the full FDDI subnet installation. The two servers would remain on the dual ring, while the five remaining SPARCStation 2's were connected to the LANplex 5012 as single attach stations. All stations retained their Ethernet connections in the unlikely event that the concentrator failed. This phase was completed on 30 September 1992. The next several sections discuss the items we considered when we developed the changeover plan.

## A. CONSIDERATIONS

Important considerations can be divided into five major categories. They are environmental, physical, link related, network availability and software compatibility. Each of these issues will be discussed in detail below.

### 1. Environmental

Synemetics had included several temperature protection schemes into the LANplex 5012. They recommend an ambient operating temperature range of 0° C to 40° C. Internally, the concentrator generates a large amount of heat, 3505 BTUs when fully configured. The first level of thermal protection is initialized at 55° C when the concentrator issues an audible alarm, a visual alarm at the panel and, optionally, sends an SMT message to the network administrator that the temperature is approaching an unacceptable level. The second level of thermal protection occurs at 70° C when the concentrator shuts itself down to prevent permanent damage to itself.

The other main environmental factor is humidity. The LANplex 5012 is designed to operate in a 5 to 95% relative noncondensing humidity range. The concentrator has no built–in facility for monitoring the moisture content in the air. In our situation, these constrains were easily met. The lab has an air conditioning system that keeps the air well within those values. If we were installing the concentrator in one of the wiring closets, these issues would have been much more important.

### 2. Physical

When the LANplex 5012 is fully loaded with expansion boards, it weighs slightly over 100 pounds. Moving it is a two person operation. It is designed to operate as a table top device or in a rack mounted configuration. Converting from table top to rack mounting is simple: add a bracket to each side of the enclosure. The LANplex 5012 cabinet remains enclosed regardless of the mounting method. It can be rack mounted to 19 or 24 inch racks.

We found that rack mounting the concentrator is a little difficult because of the weight. The user manual indicated that temporary supports for rack mounting were included. They were not shipped with the concentrator and we discovered that they were not due to be included in the near future. This meant we had to man handle the concentrator into the rack and secure it properly.

### 3. Links

The next major issue is concentrator location and cable routing. Remember from earlier discussions that the maximum ring size is 200 km and the maximum link length is 2 km using multimode fiber and standard transceivers. When a concentrator is used, each station uses an incoming link and an outgoing link. This makes the segment length twice the actual cable length. Each cable is actually a duplex fiber. In other words, a 10 m cable accounts for 20 m of ring length (10 m from the concentrator to the station plus 10 m from the station to the concentrator). Thus, the 200 km ring length becomes a 100 km linear cable length in a concentrator–based installation.

In our installation, this limit is not critical. CAPSNet only has seven stations. At most, we would only use approximately 280 m of the maximum ring length if all of our cables were 20 m long. We used 10 m cables to the servers and 20 m cables to the workstations which only uses 240 m of the maximum ring length. None of our links approach the 2 km limit. Large installations must be planned carefully, though.

Another important consideration is the curvature radius for bends. We had to make sure that we did not install the cables so that there were 90° bends. A sharp bend like that would introduce unnecessary losses or, in the extreme, break the fiber core. A gradual bend (6" or greater radius of curvature) does not present a problem.

### 4. Network Availability

Another major consideration was to ensure that the CAPS lab remained open and available as much as possible during the hardware conversion. Other than the time taken for installing hardware and software, and configuring the stations, the lab was available for general use. In a more general sense, we also wanted to make sure that we had a backup in case the concentrator failed.

To accomplish this, we left the previous Ethernet installation in place, but disabled it through the software drivers. In this way we ensure maximum network availability with a minimum of reconfiguration time.

#### 5. Software Compatibility

Initially, there was concern regarding the FDDI/DX board software drivers and their ability to operate correctly under SunOS 4.1.2M. Documentation which accompanied the drivers indicated that they would work properly in that environment, but the installation script provided on the tape indicated that there was a problem using SunOS 4.1.2M. The solution was to ignore the warning and proceed with the installation. Everything worked as planned.

## B. EXPERIENCE

Actually installing the concentrator was much easier than the planning and forethought that preceded it. The FDDI system is designed to be "plug and play" to the greatest extent possible. Each cable is terminated with a polarized MIC plug which mates with a MIC receptacle in only one orientation. MIC receptacles are keyed to allow only one type of plug. There are four different types of ports and the receptacles are keyed accordingly. A and B ports are for connections to the dual ring while M (master) and S (slave) ports are for concentration connections [Ref. 22: p. 103]. The keys are easily changed, but should be left alone once the final configuration is set.

After several different trials, we settled on a dual trunk ring for the servers. This means that the B port from the concentrator goes to the A port on suns5, the B port on suns5 goes to the A port on suns7 and the B port on suns7 goes to the A port on the concentrator. The advantage to this topology is that it uses only two concentrator ports instead of four. The major disadvantage is that if one station goes down, only a single attachment remains.

The workstations are connected in a M–S configuration. The M ports are on the concentrator and the S ports are on the workstations. This is the only way that they can be connected because the workstations are only SAS's. Five of the eight available master ports are used.

## C. CONCENTRATOR CONFIGURATION

One of the advantages of the Synemetics LANplex 5012 is that it is very easy to configure and very flexible. The LANplex 5012 is controlled by a Motorola 68030 32–bit microprocessor which controls all of the onboard functions. The LANplex 5012 has 12 expansion slots on an intelligent backplane. The backplane contains a VMEbus, which is responsible for communicating between the modules and the processor, three FDDI paths, three 4– or 16–Mbps Token Ring paths and three Ethernet paths. Only the FDDI paths are currently in use.

### 1. Slot Configuration

In our installation, slot 1 holds the System Processor Module (SPM). This is the module which controls all the functions of the concentrator. Slot 2 is not used but can have a redundant SPM. Slot 3 contains the FDDI Enterprise Access Module (FEAM). The FEAM holds one or two FDDI MAC's which are the traffic cops for FDDI. The MAC takes care of generating and controlling the tokens. Our FEAM has two MAC's installed. In addition to the MAC's, the FEAM also has FDDI A and B ports. Slots 4, 5, 7, 8, 9, 11 and 12 are empty. Slots 6 and 10 contain FDDI Concentrator Modules (FCM). The FCM's each have four FDDI M ports.

### 2. System Configuration

The current configuration for the concentrator is both MAC's active with MAC 1 on the primary FDDI path and MAC 2 on the secondary FDDI path. MAC 1 can be assigned to either the primary or secondary path while MAC 2 can be assigned to the secondary or local path. Two MAC's from the FEAM can not be assigned to the same path at the same time [Ref. 15: p. 7–3].

Each of the ports from the FCM can be assigned to either the primary, secondary or local path. All eight M ports are assigned to the primary path.

## D. FUTURE GROWTH

Eight slots in the concentrator are available for future expansion. If required, all eight slots could be used for FCM's to provide a total of 40 SAS's. Another expansion option is to add one or more Ethernet Express Modules (EEM) which support eight 10BaseT (UTP) ports. Each of these ports can support one or more Ethernet connected stations. If we filled up the remaining slots available, we would have 64 Ethernet ports and the original 10 FDDI ports.

When all the buildings on the campus are "wired" for FDDI, the CAPSNet concentrator can be easily added to the campus–wide backbone dual ring trunk, either through an FDDI router or through a direct connection.

Without any further expansion of the concentrator, the three unused ports can be connected to additional workstations.

# V. PERFORMANCE ANALYSIS

This chapter deals with actual, not theoretical, performance. In all cases, numerical values provided are from empirical analyses. In practical terms, there are several important factors which will guarantee the success or failure of a particular installation. In the case of FDDI, as well as Ethernet or any other network, there are certain parameters which are critical. The single most important factor is the bit error probability ($P_e$). If the $P_e$ is too high, the upper layers will need to get frames retransmitted more frequently, if it is too small, the system will be prohibitively expensive. The $P_e$ for FDDI is $2.5 \times 10^{-10}$ which is easily attainable with current optical fiber technology. As long as the signal to noise ratio (SNR) is sufficient, the required $P_e$ is attainable. Conversely, if the SNR is insufficient, the $P_e$ will tend toward certainty (1). The loss budget calculations in the following section directly address this concern.

The system throughput section addresses the actual, measured performance that I observed for both FDDI and Ethernet. That section also addresses system clock granularity verification as well as comparing the observed performance to the actual performance.

## A. LOSS BUDGET

A loss budget analysis is important for ensuring that the system will meet or exceed performance limits. In conventional Radio Frequency (RF) systems like Ethernet or Token Ring, the SNR must be large enough to support a specified $P_e$. For an optical system, the goal is the same, but the calculations are based on losses specific to the optical plant. The ANSI standard specifies $P_e$ of less than $2.5 \times 10^{-10}$ [Ref. 17: p. 93].

In the case of the CAPSnet installation, the calculations are very simple because the lab is in one room and there are no splices or wiring closets to confuse the calculations. Robert Kimball provides a detailed explanation of the different losses associated with FDDI [Ref. 18: p. 252] and I will follow his equation development and relate it to our specific installation. The reason for conducting this analysis is to determine whether or not our installation will meet the FDDI requirements.

The general form of the decision statistic is:

$$P \geq \mu_l + \mu_d + \mu_m + 2\sigma_T$$

where:

| | | |
|---|---|---|
| P | = | available power (defined as 11 dB for FDDI) |
| $\mu_l$ | = | aggregate component losses |
| $\mu_d$ | = | dispersion penalty |
| $\mu_m$ | = | system margin |
| $\sigma_T$ | = | total variance of the link loss distribution |

The first term on the right hand side of the inequality is the sum of the component losses in the link. These losses include propagation losses due to irregularities in the fiber, connector losses, splice losses, higher order mode losses (due to refraction inside to fiber), and the MIC ferrule delta (due to the difference between the precision test ferrule and a production ferrule). The equation is given below and Table 15 provides the μ values [Ref. 18: p. 253].

$$\mu_l = \mu_{co} l_o + \mu_{ci} l_i + \mu_{con} n_{con} + \mu_{sp} n_{sp} + \mu_{HO} + 2\delta$$

where:

| | | |
|---|---|---|
| $\mu_l$ | = | aggregate component loss |
| $\mu_{co}$ | = | outside plant cable loss |
| $l_o$ | = | length of outside fiber link |
| $\mu_{ci}$ | = | inside plant cable loss |
| $l_i$ | = | length of inside fiber link |
| $\mu_{con}$ | = | connector loss |
| $n_{con}$ | = | number of connectors |

| | | |
|---|---|---|
| $\mu_{sp}$ | = | splice loss |
| $n_{sp}$ | = | number of splices |
| $\mu_{HO}$ | = | Higher Order Mode loss |
| $\delta$ | = | MIC ferrule delta |

**Table 15: $\mu$ AT $\lambda_c$ = 1300 NM**

| Component | Variable | Mean Loss | Units |
|---|---|---|---|
| Outside Plant Cable | $\mu_{co}$ | 0.8 | dB/km |
| Inside Plant Cable | $\mu_{ci}$ | 1.0 | dB/km |
| Connector | $\mu_{con}$ | 0.4 | dB |
| Splice | $\mu_{sp}$ | 0.15 | dB |
| Higher Order mode loss | $\mu_{HO}$ | 0.5 | dB |
| MIC ferrule delta | d | 0.2 | dB |

If we substitute the values from the table above and the following values from our installation (our installation only uses continuous 10 m or 20 m fibers) into the equation for $\mu_l$,

| | | |
|---|---|---|
| $l_o$ | = | 0 meters |
| $l_i$ | = | 0.02 km (20 meters) |
| $n_{con}$ | = | 2 |
| $n_{sp}$ | = | 0 |

we get:

$$\mu_l = 0.8 \cdot 0 + 1.0 \cdot 0.02 + 0.4 \cdot 2 + 0.15 \cdot 0 + 0.5 + 2 \cdot 0.2$$

$$\mu_l = 1.72$$

The second term on the right hand side, $\mu_d$, is the dispersion penalty, which accounts for dispersion losses in the optical fiber. This is a function of bit rate, where $R_b$ = 125 Mbps, and of several chromatic characteristics of the LEDs used in FDDI.

The average segment length component accounts for links that consist of several spliced segments. This accounts for the bandwidth concatenation phenomena, which may cause a bandwidth *increase* in concatenated fibers over what is normally expected in a single, unbroken fiber.

The equation for dispersion penalty is:

$$\mu_d = 0.131 l^{1.38} + 0.1488 l_c^{0.5} l^{1.4}$$

where:

| | | |
|---|---|---|
| $\mu_d$ | = | dispersion penalty |
| $l$ | = | total link length, $l_o + l_i$ |
| $l_c$ | = | average segment length of spliced fibers |

If we substitute the following values into the equation for $\mu_d$ (use the same values as for $\mu_i$),

| | | |
|---|---|---|
| $l$ | = | 0.02 km (20 meters) |
| $l_c$ | = | 0.02 km (20 meters) |

we get:

$$\mu_d = 0.131 \cdot 0.02^{1.38} + 0.1488 \cdot 0.02^{0.5} \cdot 20^{1.4}$$

$$\mu_d = 0.00068052$$

The third term on the right hand side, $\mu_m$, is the system margin. It is a catch–all that allows for variations in the cable plant and a factor that compensates for timing variations between the light level at the output of the fiber and the light received at the lens on the receiver. I will use 1.0 dB because it is sufficient to cover any unexpected losses [Ref. 18: p. 254].

The final term on the right hand side, $2\sigma_T$, is the total variance of the link loss distribution and is defined as a function of the variances of the dispersion penalty and the loss distribution.

The equation for $\sigma_T$ is:

$$\sigma_T^2 = \sigma_l^2 + \sigma_d^2$$

where $\sigma_d$ is normally set at 0.09 dB [Ref. 18: p. 254].

The equation for $\sigma_l$ is:

$$\sigma_l^2 = \sigma_{co}^2 \cdot \frac{l_o^2}{R} + \sigma_{ci}^2 \cdot l_i^2 + \sigma_{con}^2 \cdot N_{con} + \sigma_{sp}^2 \cdot N_{sp}$$

where:

| | | |
|---|---|---|
| $\sigma_l$ | = | standard deviation of the loss distribution |
| $\sigma_{co}$ | = | outside plant cable loss standard deviation |
| $l_o$ | = | length of outside fiber link |
| R | = | number of cable segments in the link |
| $\sigma_{ci}$ | = | inside plant cable loss standard deviation |
| $l_i$ | = | length of inside fiber link |
| $\sigma_{con}$ | = | connector loss standard deviation |
| $N_{con}$ | = | number of connectors |
| $\sigma_{sp}$ | = | splice loss standard deviation |
| $N_{sp}$ | = | number of splices |

The $\sigma$ values associated with each of the variables in the previous equation are given in Table 16 [Ref. 18: p. 254].

**Table 16: STANDARD DEVIATION OF LOSS CHARACTERISTICS**

| Component | Variable | Standard deviation | Units |
|---|---|---|---|
| Outside Plant Cable | $\sigma_{co}$ | 0.25 | dB/km |
| Inside Plant Cable | $\sigma_{ci}$ | 0.0 | dB/km |

**Table 16: STANDARD DEVIATION OF LOSS CHARACTERISTICS (Continued)**

| Component | Variable | Standard deviation | Units |
|---|---|---|---|
| Connector | $\sigma_{con}$ | 0.2 | dB |
| Splice | $\sigma_{sp}$ | 0.1 | dB |

When we substitute the values from the table above and the l values given earlier, we get:

$$\sigma_l^2 = 0.0625 \cdot \frac{0}{1} + 0.0 \cdot 0.0004 + 0.04 \cdot 2 + 0.1 \cdot 0$$
$$\sigma_l^2 = 0.08$$

Now that we know $\sigma_d$ and $\sigma_l$, we can substitute them into the equation for $\sigma_T$ and solve it, which yields:

$$\sigma_T = 0.2968$$

The final step is to substitute all the intermediate results back into the original equation to verify that we have not exceeded the loss budget. When we substitute in, we get:

$$11 \geq 1.72 + 0.00068052 + 1 + 2 \cdot 0.2968$$
$$11 \geq 3.3143$$

If the right hand side of the equation exceeded 11 dB, we would need to go back to our installation and figure out where we could improve the loss budget. The area that would provide the greatest change with the least effort would be the aggregate loss factor. Two ways to improve that factor would be to shorten the links between transmitter and receiver or reduce the number of connectors.

For comparison purposes, the link losses for various other link lengths are given in Table 17.

**Table 17: LOSS BUDGET FOR VARIOUS LINK LENGTHS**

| Link length | Loss (dB) |
|---|---|
| 10 m | 3.30388 |
| 20 m | 3.31431 |
| 30 m | 3.32486 |
| 50 m | 3.34623 |
| 75 m | 3.37339 |
| 100 m | 3.40097 |

In all cases, we have more than 7 dB of signal excess. When CAPSnet is connected to the CS ring or the campus ring, there may be additional demands and losses. T. McIntosh addresses specific wiring considerations for buildings and campuses [Ref. 31: pp. 242–250].

## B. SYSTEM THROUGHPUT

Theoretically, networks can approach 100% transmission efficiency, but there are certain trade–offs that must be addressed. Contention–based protocols which approach 100% transmission efficiency have excessive wait delays associated with them. Collision–free protocols are better suited for approaching the transmission efficiency limit.

### 1. Clock Accuracy Verification

Timing analysis is critical to determining how well the system performs over the network. Recent studies have shown that bottlenecks in the protocol stacks and the processors are more detrimental to network speed than the raw data transfer rate. In order to determine how well the protocols performed, I needed to be able to time different data transfers and compare them.

I used C test programs, rather than Ada, to perform the timing tests because all of the SunOS calls are in C. Even though the SunOS manuals discuss a 1 μsec clock resolution [Ref. 13: pp. 24–25], they advise caution in accepting μsec values as valid [Ref. 12: p. 761]. In other words, the μsec clock may not track properly when compared to the NBS (National Bureau of Standards) atomic clocks. For hard real–time systems where absolute temporal triggering *and* procedure duration are critical, this would present a major challenge. By contrast, my tests were relatively short and did not require an absolute time reference.

Previous timing studies conducted using a UNIX or system call for time only achieved a resolution of approximately 20 msec. For my purposes, I needed a finer resolution than this. To ensure that I was actually able to achieve a 1 μsec resolution–or, more accurately–comfortably accept a 1 μsec resolution, I used the test program shown in Appendix A. I ran it in two different configurations. The first was with 100 iterations and the second was with 1000 iterations. The concept is simple: call `gettimeofday()` and store the results in an array. The procedure call passes in two pointers to structures which tells the procedure where to place the results. The results are copied into an array for subsequent display.

The structures represent absolute time and time zone. The time structure, called `timeval`, consists of two long integers and the time zone structure, called `timezone`, consists of two integers. I will only discuss `timeval`, because there was no need for `timezone`.

The two components of `timeval` are `tv_sec` and `tv_usec`, which represent the elapsed seconds and microseconds since 00:00 GMT 1 Jan 1970 [Ref. 12: p. 760]. I stored the results of `tv_usec` in the array to verify the clock timing. After the array is full, the program displays the results on the monitor.

The results showed that I could expect 1 μsec clock resolution within reason. Every 10 msec or so, the UNIX pager or swapper preempts the application for approximately 120 μsec. No applications can preempt the pager and swapper even by altering their "niceness" (or priority) levels. I executed the program (1000 iteration version) without adjusting the nice value and with a nice rating of -20, which is the highest value allowed without altering the code in the kernel. I found that there were no significant differences in execution delays between the two settings.

The discussion up to this point has centered around determining the system clock resolution. The reason for doing that is to be able to accurately determine how long data transfers from one point to another will take. Inaccurate timing would jeopardize the validity of the data I collected. The next section will discuss the test procedure in detail and provide an analysis of the results.

### 2. Timing Test Procedure

The procedure used to find network file transfer speed was to time a remote file copy (**rcp**) from one machine to another. Since we were interested in how quickly data moved from one place to another, I performed four sets of 21 different tests. The first and second sets were conducted on Ethernet while the third and fourth sets were conducted on FDDI. The first set of tests was a baseline and was conducted under normal circumstances, which simulated normal network activity. The second set of tests was conducted under "ideal" conditions with all transfers within the subnet and no other network traffic. The third set was a repeat of the first and the fourth set was a repeat of the second.

Several variations of the C programs were attempted until I found the one that provided optimum performance. Appendix B shows the actual program to accomplish the file transfers. This is where the distinction between TCP and UDP as the upper protocol layer makes a tremendous difference. UNIX (at least Sun OS 4.1.1 UNIX) uses UDP as the underlying protocol for local network copying. In other words, when a user uses the UNIX **cp** command to copy a file from one location to another, the UNIX kernel sets it up as a UDP transfer. As explained earlier, UDP is an unreliable, connectionless service and, as a result, the UNIX kernel forces a write to the hardware (usually a hard disk) before it will recognize that the task is completed. This adds additional overhead and is clearly not the file transfer method to use to measure network performance. A better method of measuring network performance is to use an NFS **get** command or the UNIX **rcp** command. The major difference between these commands and the **cp** command is that they use TCP. Since TCP is a reliable transport service, the UNIX kernel recognizes that the task is complete when the protocol stack sends the ACK. This process takes much less time because it does not depend upon the physical transfer to the storage medium, although at some point the receive buffers must be moved to a storage device before overflowing.

Each set of tests consisted of three groups of file transfers. The test files were selected based on size. The criteria for size selection is described in the following paragraphs.

The largest one had to be larger than 256 kilobytes so that it would exceed the size of the buffers on the interface cards. I selected a file slightly larger than 1 Mbyte (1155959 bytes) in size to minimize the effects (by percentage) of overhead.

The next file had to be smaller than 256 kilobytes, but larger than the size of an 802.3 or FDDI packet. The space reserved for data is 1500 bytes in an 802.3 packet and 4478 bytes in an FDDI packet. I selected a file slightly larger than 56 kilobytes (58080 bytes) in size to minimize the effects (by percentage) of overhead.

The final file had to be smaller than the minimum size for an 802.3 packet. This was to ensure that the smallest possible packet was sent. I selected an 11 byte file to simulate the results of a computation being passed back to a caller. The minimum packet size for an 802.3 network is 46 bytes which ensures that it will take longer than $2\tau$ to transmit it. FDDI has no such minimum. Table 18 shows the percentages of overhead for each of the different file sizes and transmission media. Percentage of overhead is calculated by dividing the number of bytes of overhead by the number of data bytes, then multiplying the result by 100. The overhead for the small file transfer is huge when compared to the data being transferred. This is a result of TCP, IP and the physical layer adding headers and trailers to the data they each receive.

**Table 18: PERCENTAGE OF OVERHEAD**

| File Size | Ether et | FDDI |
|---|---|---|
| Large (1155959 bytes) | 4.81 | 1.63 |
| Medium (58080 bytes) | 4.82 | 1.74 |
| Small (11 bytes) | 636.36 | 654.55 |

Each file was transferred between seven pairs of nodes and each transfer was performed 500 times to provide a statistically significant sample. Luckily, the file transfers only took a short time to accomplish.

### a. Test Set One

Test Set One was conducted from sun51 (see Appendix E for a notional network diagram). The tests were done in this order: sun51 to sun52, sun51 to suns5, sun51 to suns7, suns5 to suns7, suns7 to suns5, suns5 to sun51 and suns7 to sun51. As explained above, each transfer was performed 500 times. The test was conducted on the CAPS subnet to avoid bridge latency (at sun53) for both the command and the response. The subnet was in use for normal CAPS development. This test provides "typical" results for a user who was working on the CAPS subnet, either on campus or dialed in remotely.

#### *b. Test Set Two*

Test Set Two was conducted from sun51. The tests were done in the same order as before: sun51 to sun52, sun51 to suns5, sun51 to suns7, suns5 to suns7, suns7 to suns5, suns5 to sun51 and suns7 to sun51. As explained above, each transfer was performed 500 times. This test was designed to test "ideal" file transfers. The only traffic on the net was that generated by the tests. All other users had logged off.

#### *c. Test Set Three*

Test Set Three was conducted from sun51 (see Appendix F for a notional network diagram). The tests were done in this order: sun51 to sun52, sun51 to suns5, sun51 to suns7, suns5 to suns7, suns7 to suns5, suns5 to sun51 and suns7 to sun51. As explained above, each transfer was performed 500 times. Again, the test was conducted on the CAPS subnet to avoid bridge latency for both the command and the response. In this case the bridge latency would be larger because of the conversion from FDDI packets to Ethernet packets. Comparing the two sets of results would have been more difficult because of the extra variable.

#### *d. Test Set Four*

Test Set Four was conducted from sun51. The tests were done in this order: sun51 to sun52, sun51 to suns5, sun51 to suns7, suns5 to suns7, suns7 to suns5, suns5 to sun51 and suns7 to sun51. As explained above, each transfer was performed 500 times. This test was designed to test "ideal" file transfers. The only traffic on the net was that generated by the tests. All other users had logged off.

### 3. Timing Test Results

The general results of the timing tests are provided in the following sections. Graphical results for each test are provided in Appendix G. These results only give an average of the time taken for each file transfer. A more detailed analysis of the distributions is provided in the Distribution of Message Delays section. In order to make the results more meaningful and reduce the disparities between the different machines, I grouped the transfers by type. Group 1 transfers were those from servers to workstations, Group 2 transfers were those from workstations to servers, Group 3 transfers were those from workstation to workstation and Group 4 transfers were those from server to server. Further, this type of grouping allows the most common types of file transfers to be more easily analyzed. In the CAPS environment, the most common data transfers are between workstations and servers.

#### *a. Test Set One*

Table 19 shows the results from performing the Ethernet file transfer test with normal net loading. It is presented by the groups mentioned in the previous section. As expected, the larger the file, the longer it took to complete the file transfer. Another interesting point is that the medium file transfer did not take much more time to complete than the small file transfer.

**Table 19: AVERAGE TIME IN SECONDS FOR ETHERNET UNDER NORMAL LOAD**

| File size | Group #1 | Group #2 | Group #3 | Group #4 |
|---|---|---|---|---|
| Large file (1155959 bytes) | 4.57111 | 4.54401 | 4.50775 | 4.6884 |
| Medium file (58080 bytes) | 3.64403 | 3.60362 | 3.43904 | 3.64839 |
| Small file (11 bytes) | 3.50187 | 3.52319 | 3.43241 | 3.59244 |

Data transfer rates are consistent across file sizes. Table 20 shows the same data from Table 19 converted to data transfer rate instead of raw transfer times.

The server to server transfers (group 4) were always the slowest, workstation to workstation transfers (group 3) were the fastest, and workstation to server transfers (group 2) and server to workstation transfers (group 1) were in the middle with the slightly better performance available from the workstation to server transfers.

**Table 20: DATA TRANSFER RATES IN MBPS FOR ETHERNET UNDER NORMAL LOAD**

| File size | Group #1 | Group #2 | Group #3 | Group #4 |
|---|---|---|---|---|
| Large file (1155959 bytes) | 2.02307 | 2.03513 | 2.05151 | 1.97246 |
| Medium file (58080 bytes) | 0.12751 | 0.12894 | 0.13511 | 0.12735 |
| Small file (11 bytes) | 2.5E-05 | 2.5E-05 | 2.6E-05 | 2.4E-05 |

### *b. Test Set Two*

Table 21 shows the results from performing the Ethernet file transfer test with no net loading. It is presented by the groups mentioned in the previous section. As expected, the larger the file, the longer it took to complete the transfer. The comments from the previous section apply equally well here.

**Table 21: AVERAGE TIME IN SECONDS FOR ETHERNET UNDER NO LOAD**

| File size | Group #1 | Group #2 | Group #3 | Group #4 |
|---|---|---|---|---|
| Large file (1155959 bytes) | 4.61714 | 4.60216 | 4.42663 | 4.70986 |
| Medium file (58080 bytes) | 3.63165 | 3.84411 | 3.59872 | 3.68031 |
| Small file (11 bytes) | 3.5717 | 3.52192 | 3.39746 | 3.52963 |

Data transfer rates are consistent across file sizes. Table 22 shows the same data from Table 21 converted to data transfer rate instead of raw transfer times.

In all cases, the workstation to server transfers (group 2) were fastest and the server to workstation transfers (group 1) were always the slowest.

Table 22: DATA TRANSFER RATES IN MBPS FOR ETHERNET UNDER NO LOAD

| File size | Group #1 | Group #2 | Group #3 | Group #4 |
|---|---|---|---|---|
| Large file (1155959 bytes) | 2.0029 | 2.00942 | 2.0891 | 1.96347 |
| Medium file (58080 bytes) | 0.12794 | 0.12087 | 0.12911 | 0.12625 |
| Small file (11 bytes) | 2.5E-05 | 2.5E-05 | 2.6E-05 | 2.5E-05 |

### c. *Test Set Three*

Table 23 shows the results from performing the FDDI file transfer test with normal net loading. It is presented by the groups mentioned in the previous section. The results are not as clear cut as are those from the previous two tests. The servers no longer posted the slowest transfer times primarily due to the microprocessor dedicated to packetizing the FDDI data and controlling the interface.

Table 23: AVERAGE TIME IN SECONDS FOR FDDI UNDER NORMAL LOAD

| File size | Group #1 | Group #2 | Group #3 | Group #4 |
|---|---|---|---|---|
| Large file (1155959 bytes) | 3.96714 | 4.18395 | 3.99961 | 4.0783 |
| Medium file (58080 bytes) | 3.21534 | 3.20085 | 3.23287 | 3.18365 |
| Small file (11 bytes) | 3.39959 | 3.39859 | 3.40128 | 3.3994 |

In this case, data transfer rates are not consistent across file sizes. The small file transfers actually took longer to complete than the medium files. This is probably an anomaly since there is no logical reason for the small files to take more time than the medium files. This may be the result of daily backups which may have been conducted during the tests. Table 24 shows the same data from Table 23 converted to data transfer rate instead of raw transfer times.

Table 24: DATA TRANSFER RATES IN MBPS FOR FDDI UNDER NORMAL LOAD

| File size | Group #1 | Group #2 | Group #3 | Group #4 |
|---|---|---|---|---|
| Large file (1155959 bytes) | 2.33107 | 2.21027 | 2.31214 | 2.26753 |
| Medium file (58080 bytes) | 0.14451 | 0.14516 | 0.14372 | 0.14595 |
| Small file (11 bytes) | 2.6E-05 | 2.6E-05 | 2.6E-05 | 2.6E-05 |

### *d. Test Set Four*

Table 25 shows the results from performing the FDDI file transfer test with no net loading. It is presented by the groups mentioned in the previous section. As with the previous test set, The server to server transfers were no longer the slowest. The small and medium file transfers were fastest, but the large file was next to the slowest. This is probably due to lack of buffer space on the interface card.

Table 25: AVERAGE TIME IN SECONDS FOR FDDI UNDER NO LOAD

| File size | Group #1 | Group #2 | Group #3 | Group #4 |
|---|---|---|---|---|
| Large file (1155959 bytes) | 4.04894 | 3.20263 | 3.18897 | 3.76448 |
| Medium file (58080 bytes) | 3.19917 | 3.40358 | 3.22202 | 3.18418 |
| Small file (11 bytes) | 3.41572 | 3.23914 | 3.39594 | 3.19937 |

Data transfer rates are not entirely consistent across file sizes. The same anomaly exists here for the same reasons. Table 26 shows the same data from Table 25 converted to data transfer rate instead of raw transfer times.

Table 26: DATA TRANSFER RATE IN MBPS FOR FDDI UNDER NO LOAD

| File size | Group #1 | Group #2 | Group #3 | Group #4 |
|---|---|---|---|---|
| Large file (1155959 bytes) | 2.20105 | 2.03086 | 2.29328 | 2.02035 |
| Medium file (58080 bytes) | 0.12832 | 0.12902 | 0.13669 | 0.12076 |
| Small file (11 bytes) | 2.4E-05 | 2.6E-05 | 2.6E-05 | 2.6E-05 |

## 4. Modeling Software Analysis

I used the CACI Products Company simulation software LANNET II.5™ to model file transfers between workstations and servers. I set up two classes of simulations: one for Ethernet and one for FDDI. The simulations were of Test #2 and Test #4 (Ethernet and FDDI no load performance, respectively). Adding net loading would have been fairly easy because LANNET II.5™ supports 12 different kinds of distributions and provides the ability to make user–defined distributions.

I attempted to include the TCP/IP overhead in the LANNET II.5™ simulation by adjusting the overhead associated with the file transfers. Otherwise, standard network parameters were used for the simulations. The following two sections provide a summary of the simulation results. The detailed simulation report is provided in Appendix D.

### *a. Ethernet*

A comparison of the file transfer times that I computed manually and those computed by LANNET II.5™ are shown in Table 27. The LANNET II.5™ times are smaller than those I calculated manually because the LANNET II.5™ times do not include the TCP/IP acknowledgments for the sliding window. Neither of these sets of times include the processing overhead (which can be added to the simulation) or the file access times for the hard drives (which can also be added to the simulation).

**Table 27: COMPARISON OF FILE TRANSFER TIMES IN μS**

| File size | LANNET II.5™ | Manual |
|---|---|---|
| Large (1155959 bytes) | 988596.000 | 1028626.4 |
| Medium (58080 bytes) | 49678.301 | 51660.8 |
| Small (11 bytes) | 85.600 | 170.4 |

These data suggest that it is reasonable to expect data transfer times close to those shown if the data is instantaneously available to the interface card at the beginning of the file transfer. In other words, it is not reasonable to expect data transfer times close to these under normal circumstances.

### *b. FDDI*

A comparison of the file transfer times that I computed manually and those computed by LANNET II.5™ are shown in Table 28. The LANNET II.5™ times do not exactly match those I calculated manually, but they are close enough to validate my values. Neither of these sets of times include the processing overhead (which can be added to the simulation) or the file access times for the hard drives (which can also be added to the simulation).

**Table 28: COMPARISON OF FILE TRANSFER TIMES IN μS**

| File size | LANNET II.5™ | Manual |
|---|---|---|
| Large (1155959 bytes) | 94419.234 | 94318. |
| Medium (58080 bytes) | 4745.384 | 4764. |
| Small (11 bytes) | 8.136 | 6.32 |

The question that remains is, "Why do the actual values and the theoretical (or simulation) values differ so greatly?" Obviously, there are other factors which I did not address with the theoretical calculations and were not included in the LANNET II.5™ simulation. I will address the missing bottlenecks in the following section.

## C. BOTTLENECK IDENTIFICATION

Not surprisingly, I found that there was a significant difference between the theoretical file transfer time and the actual file transfer time for both Ethernet and FDDI. Overall, there was not a one–order–of–magnitude improvement in the performance of FDDI over Ethernet. Three areas in the computer directly affect the difference. First, the interface hardware has a maximum throughput which may be much slower than the network data rate. Second, the data has to be processed in one way or another and that processing is not completed instantaneously. Third, once the data is processed, it must be moved to some form of permanent or semi–permanent storage.

### 1. Interface Hardware

The actual hardware interface operates at 125 MHz (for FDDI). All of the hardware that converts the incoming serial stream to a parallel stream for the microprocessor operates at that speed as well. Once the data gets into the buffers, the controlling factor becomes the microprocessor. In order to keep up with the incoming bit stream, the microprocessor would have to use a 100 MHz clock (which is generally not available for production equipment) if it could only read data one byte at a time. Data bytes get shifted into the buffers at a 12.5 MHz rate (100 Mbps + 8bits/byte). The interface would be easy if the microprocessor could read the buffers in one clock cycle, but it cannot. The MC68020 that is used in the Sun FDDI/DX interface typically takes eight clock cycles to read a byte of data from memory

[Ref. 16: pp. 10-2–10-3]. There are certain economies of scale, though. It takes the same amount of time to read four bytes of data from memory as it takes to read one byte, provided that the interface is built around a 32–bit data bus. This means is that the slowest clock rate for the microprocessor is 25 MHz.

While the majority of the decoding and shifting functions are performed by dedicated chips, some functions require the processor. One of those functions is forwarding the data to the main processor. Moving the data from the buffer to main memory takes at least 16 clock cycles [Ref. 16: p. 10-2] once the microprocessor has seized control of the VME bus.

The preceding paragraphs have only discussed memory reads and writes and not the other functions that the interface microprocessor must also perform. It must also generate and respond to interrupts for or from IP. One instance is when the incoming file is larger than the 256 kilobyte buffer and the buffer overflows.

In the macro sense, interrupt handling takes much longer than simple memory reads and writes. It is easy to see that the interface is one source that may reduce the data transfer rate.

In my estimation, the overall delay effect of the interface card is minimal because in most common networking environments, the size of the data to be transferred is smaller than the buffer size. Also, processor features on the MC68020 like prefetching make it more efficient. A 25 MHz clock rate would probably be sufficient to keep up with the incoming data stream. The most likely place for a bottleneck would be the communications between the interface card and the main processor. The amount of time it takes to seize the main bus to accomplish the transfer depends on the interrupt level. Overall, the delay should not exceed 0.75 µsec if we assume that at most it would take 30 clock cycles to complete the current processor instruction and respond to the interrupt. The only time that it would take longer to seize the bus is when the processor is currently handling a higher level interrupt.

## 2. Protocol Stacks

Once the data is in main memory, the main processor calls IP and IP strips the header and calls TCP. TCP then strips its own header and passes the data to the appropriate process. The way that this is actually accomplished is that the main processor updates the pointer to the beginning of the data segment by adding the offset in the HLEN field. It also calculates the end of the IP datagram by adding the value in the TOTAL LENGTH field to the original pointer. TCP strips the header from its datagram by updating the beginning pointer. It then passes a pointer and length or two pointers to the appropriate destination process. In addition to the pointer manipulation, TCP and IP must also take care of other tasks like comparing header checksums and sending ACKs.

TCP must also decide where to put the information for the appropriate process based on the incoming traffic. In some cases the information is in a lookup table and in others, it must compute offsets based on internal constants and the destination port value in the header. All of this takes place under software control, which is subject to suspension because of higher priority tasks, *i.e.* UNIX pager and swapper.

As shown in the previous section, the workstation to workstation file transfers are fastest while server to server transfers are slowest, and the other transfers were in between the two extremes. The reason transfers involving the servers take longer is that the servers do not process the data as quickly as the workstations. The primary reason is that the multiprocessing software on the servers adds additional overhead. While conducting the tests to verify that I could read the system clock to a 1 μsec resolution, I noticed that the loop consisting of checking a counter, updating the counter, making a subroutine call, returning from the subroutine call and storing the results of the subroutine call in an array took about 27.5 μsec on a workstation. The same process on the servers took about 33.3 μsec, which represents a performance penalty of about 21%.

The TCP, IP and network elapsed time can be described graphically as shown in Figure 5.1 [Ref. 39: p. 45]. T represents the total time involved in the data transfer and the t's represent the time taken to complete a function in the protocol stack. The $t_T$ value represents the TCP portion of the protocol stack, the $t_I$ value represents the IP portion of the protocol stack, the $t_P$ value represents the physical transfer protocol in use (Ethernet or FDDI, in our case) and $t_C$ value represents the time in the channel. If all other values remain the same, decreasing $t_C$ by a factor of 10 would change T by some fractional amount. The amount of the change depends upon the processing time and other overhead associated with the transfers.

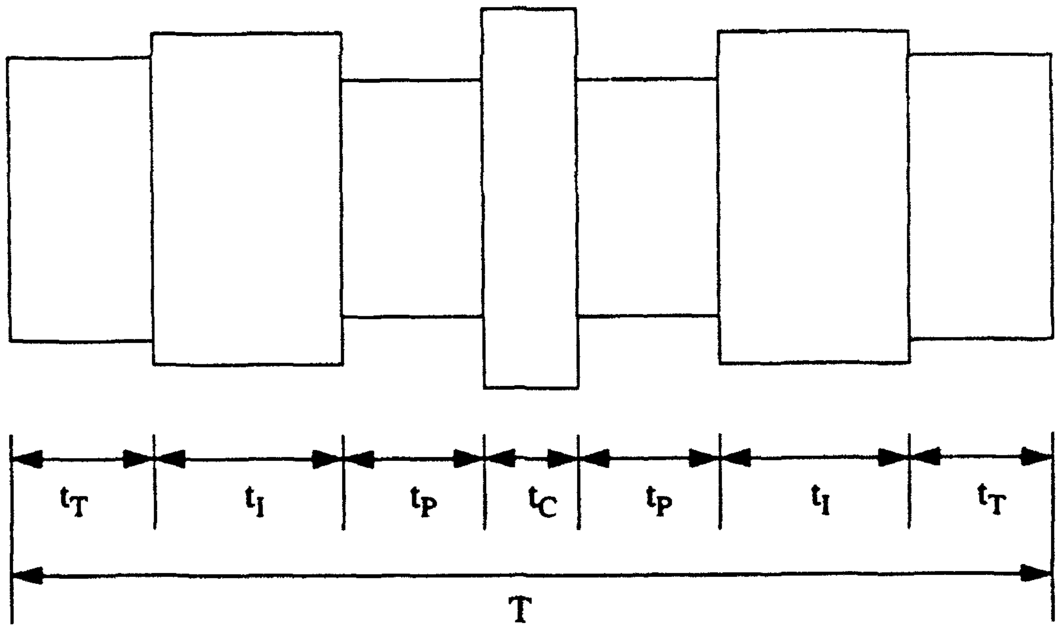


**Figure 5.1 Protocol stack**

If we assume that the $t_c$ values from Table 27 and Table 28 are correct, we can compute the time taken by the stack and the storage systems. The transfer times also include the time taken to establish the connection and close it again. Establishing the connection requires a three way handshake [Ref. 3: pp. 194–195] which translates to a minimum size frame in each direction before starting the data transfer. The third part of the handshake can be piggybacked on the first data frame. Closing the connection uses a modified three way handshake [Ref. 3: pp. 196–197] but it imposes no additional penalty because the close instruction can be piggybacked on the last data frame and the response can be piggybacked on the ACK for the last frame.

We can subtract the $t_c$ values from the total time taken and then use a minimum mean square error formula and linear regression to determine how much time is associated with the processing the bit stream (variable) and how much is taken up by overhead (fixed). The fixed and variable values are shown in Table 29.

**Table 29: FIXED AND VARIABLE TIMES (IN SECONDS) FOR ETHERNET AND FDDI TRANSFERS**

| | Ethernet | FDDI |
|---|---|---|
| Variable | 1.1626E-08 | 7.6474E-07 |
| Fixed | 3.57530133 | 3.68333285 |

The variable times represent seconds per byte. Since the actual data transfer time was subtracted out, these values include time taken to establish the connection, read the data, encapsulate in three layers of protocols and restore the data at the other end. A large part of the fixed time is probably related to building the connection because it involves manipulating connection ports and updating routing tables. Without access to the source code for the establish connection procedure and the build packet procedures it is difficult to determine how much is involved in each one.

Based on hardware considerations with respect to the interface card and the observations I made with respect to hard drive access, I believe that the greatest percentage of overhead is associated with building the TCP/IP connection and processing the data through the protocol stacks.

### 3. Storage

The storage subsystem is one of the slowest components in a computer system. At the fast end of the spectrum are the IPI and SCSI drives, next come tape backup units, then erasable optical systems and finally CD–ROM and floppy drives. The main reason that storage subsystems are so slow is because they are primarily mechanical devices.

In the tests that I performed, data had to be read from one disk and written to another. Even though the data transfer rate from the controller to the hard drive is between 3.5 and 4.2 Mbps, actual data transfers must also account for seek and settle times for the read/write heads. These times are in the millisecond range, typically between 9.5 and 15 milliseconds. This adds substantial overhead to the data storage process.

In order to try to find out how long a disk read/write cycle actually takes, I duplicated the three files on to the same device and timed the execution using the same calls that I used for the file transfers. The results were quite surprising and are shown in Table 30. Notice that the time to duplicate the large file took longer than the time required to transfer that file to a remote storage device.

**Table 30: DUPLICATE FILE TIMES**

| File size | Elapsed time (s) |
|---|---|
| Large (1155959 bytes) | 11.052 |

Table 30: DUPLICATE FILE TIMES (Continued)

| File size | Elapsed time (s) |
|---|---|
| Medium (58080 bytes) | 0.5144 |
| Small (11 bytes) | 0.1679 |

I felt that the times from this test were unreliable and not representative of the tests I was conducting. A more reasonable method may be to use the synchronous data transfer rates provided by the UNIX **dmesg** command. After reviewing the raw data, I noticed that the first file transfer took longer than those following it. I believe that this is due to caching which would allow the system to use the cache instead of reading the data from the hard drive again. Continuing with that assumption, the read time for the file transfer get averaged over 500 iterations and accounts for an insignificant amount of the values presented here.

If a user was sending different large files to remote stations, the effect would be noticeable, but sending the same file to several locations would pose no additional overhead. Storage access and transfer rates do not apply when the data being transferred is RAM resident.

## D. ERROR RATE

With FDDI, we can expect a bit error rate of less than $2.5 \times 10^{-10}$ because of the analysis in section A. If we experienced large numbers of errors, the most probable causes would be in the optical fibers themselves or in the optical transceivers. The functionality built into the concentrator and the SMT services from FDDI would allow us to quickly isolate the station or link causing the problem and correct it.

With Ethernet, the situation is quite different. In the normal configuration, Ethernet cabling is susceptible to RF interference. Coaxial cable is less affected by RF energy than UTP, but it does not provide the same expectation of signal clarity that optical fibers do. Bit error probabilities in the range of $1 \times 10^{-8}$ would be reasonable and single errors or short error bursts will be corrected, or at least detected, by the CRC. Isolating the station causing the problem is much more involved because Ethernet does not support the network management capabilities that FDDI does. Further, the medium is broadcast, rather than a composite of point–to–point links.

If the Physical Layer entity can not correct the corrupted data streams, the upper layers will take care of it. In our case, TCP would send a NAK for the appropriate frame and it would be resent.

## E. DISTRIBUTION OF MESSAGE DELAYS

In theory, the distribution of message delays is a continuous random process. In our application, the distribution is discrete because the system clock is only accurate to one microsecond. When viewed over a two to five second window, the distribution appears nearly continuous. The message delivery distribution for Ethernet takes the general form of a Rayleigh probability density function. The Rayleigh function is asymmetrical and has the largest probability density on the left side and continues to infinity on the right. Even though the Ethernet file transfers are modal in nature, the envelope of the distribution plot is still a Rayleigh function. The mathematical expression for a Rayleigh function is:

$$f = h\frac{r_0}{\sigma^2} \cdot e^{(-r_0^2)/(2\sigma^2)}$$

where $\sigma$ is the variance and $h$ is a height adjustment multiplier. The function is only defined for $r_0 \geq 0$. The plot of a Rayleigh function for $\sigma = 1$and $h = 1$ is shown in Figure 5.2. This is appropriate for Ethernet because, as I mentioned earlier, the majority of the transfers take place relatively fast, but there is a finite possibility that a transfer will take an infinite amount of time.

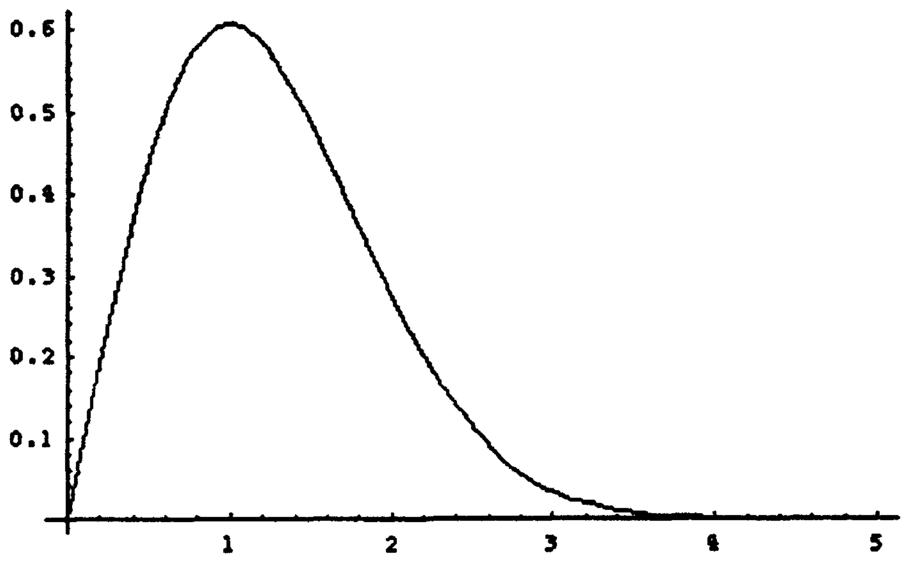


**Figure 5.2 Rayleigh distribution function**

To calculate the probability density functions, I took the samples that I had collected and performed some basic statistical analyses (mean, standard deviation). I then grouped the data into four categories based on the type cf file transfer that I attempted. The four groups were: server to workstation, workstation to server, workstation to workstation and server to server. I did this to get a larger sample and reduce the effects of a particular machine.

From there, I generated a histogram and plotted it. A sample of the actual data is provided in Figure 5.3. A complete set of the histogram plots is shown in Appendix G. After I generated the plot, I used *Mathematica* to graph the ideal function with the same parameters. Figure 5.4 shows a sample generated to match Test #1 (normal network load), Group #1 (servers to workstations).

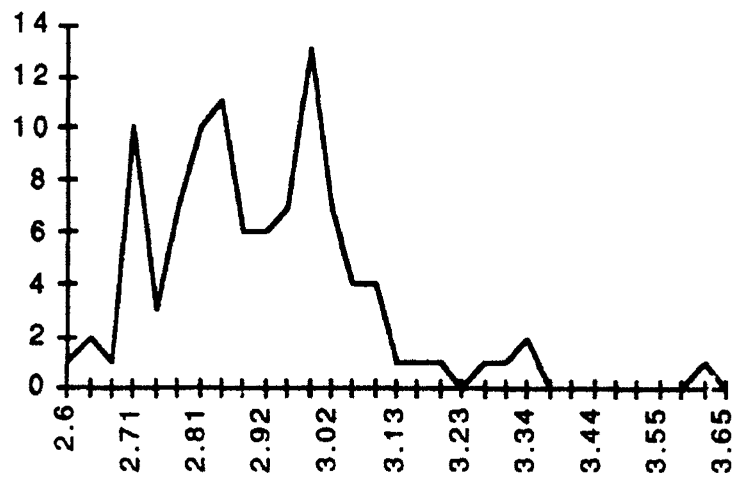


**Figure 5.3 Actual data**

Plots are not provided for the Rayleigh functions because they look the same with the exception of the height and spread. I have summarized the results of the plots in Table 30.

I did not use Rayleigh distributions to model the FDDI delay distributions because they are not stochastic processes. Their histogram plots are also provided in Appendix G. One major item to notice is that the FDDI plots are modal in nature. There are two or more definite spikes which represent the effect of TRT and THT. Figure G.36 and Figure G.47 show this feature best.

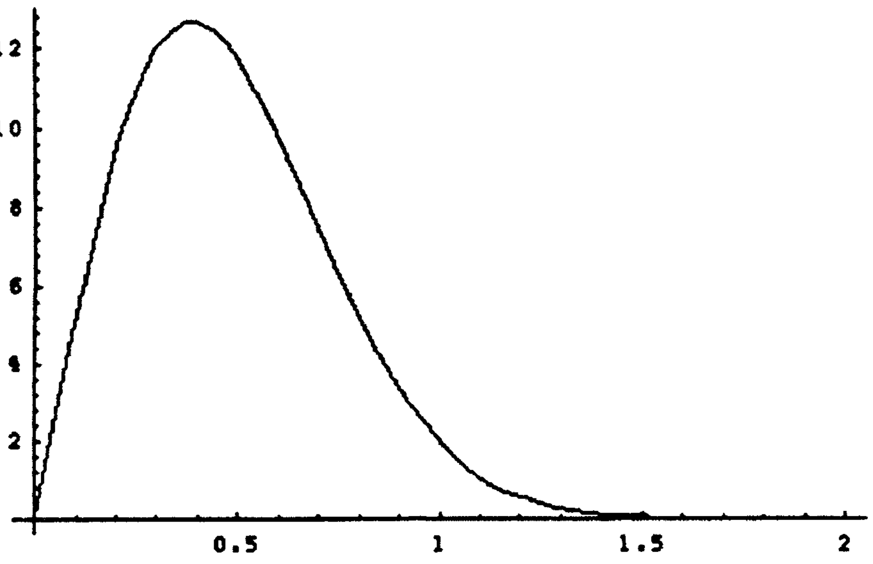


**Figure 5.4 Model for actual data in Figure 5.2**

**Table 31: VALUES OF σ AND *h* FOR RAYLEIGH DISTRIBUTION**

| | | | σ | *h* |
|---|---|---|---|---|
| Test #1 | Trial #1 | Group #1 | 0.15 | 8.1 |
| | | Group #2 | 0.24 | 12 |
| | | Group #3 | 0.19 | 8.2 |
| | | Group #4 | 0.55 | 30 |
| | Trial #2 | Group #1 | 0.12 | 8.2 |
| | | Group #2 | 0.12 | 15 |
| | | Group #3 | 0.02 | 0.05 |
| | | Group #4 | 0.19 | 8.2 |
| | Trial #3 | Group #1 | 0.8 | 40 |
| Test #1 | Trial #3 | Group #2 | 0.19 | 14 |
| | | Group #3 | 0.63 | 25 |
| | | Group #4 | 0.19 | 15 |

**Table 31: VALUES OF σ AND *h* FOR RAYLEIGH DISTRIBUTION (Continued)**

| | | | σ | *h* |
|---|---|---|---|---|
| Test #2 | Trial #1 | Group #1 | 0.68 | 20 |
| | | Group #2 | 0.65 | 15.8 |
| | | Group #3 | 0.15 | 4.5 |
| | | Group #4 | 0.47 | 11.1 |
| | Trial #2 | Group #1 | 0.39 | 20 |
| | | Group #2 | 0.36 | 21.6 |
| | | Group #3 | 0.19 | 6.4 |
| | | Group #4 | 0.23 | 7.8 |
| | Trial #3 | Group #1 | 0.38 | 15.5 |
| | | Group #2 | 0.23 | 10.2 |
| | | Group #3 | 0.20 | 6.4 |
| | | Group #4 | 0.22 | 13 |

# VI. CONCLUSIONS

The results that I recorded were much different than what I expected. Even though I knew that the different types of processing would take some time, I did not have a feel for how much time they actually took. The next three sections provide a summary of the results that I expected, the results that I observed and a comparison of the two. The final section discusses topics for further study.

## A. EXPECTED RESULTS

When I first started this thesis, I expected the FDDI net to have significantly faster data transfer rates than the Ethernet. Although I did not expect to see data transfer rates that were ten times faster, I did expect to see data transfer rates that were three to five times faster. The reason I felt that this was appropriate was because, theoretically, FDDI is approximately 20 times faster than Ethernet as far as raw data transfer speed. Remember that Ethernet has a maximum data transfer rate on the medium of approximately 5.2 Mbps. FDDI, on the other hand, does transfer data at 100 Mbps because there is no time lost to collision processing.

The big problem with the preceding discussion is that it does not take into account other factors such as net loading and the number of datagrams actually queued and ready for transmission. Maximum network utilization occurs when every station has traffic to send when it senses a clear channel or when it receives the token. My tests, on the other hand, were more of a quiescent state test rather than a steady state test.

I also expected the transfers between the servers to be much faster than transfers between any of the other stations–at least for the Ethernet tests. I felt that the multiprocessors could improve performance because one processor could perform the kernel operations while the other handled network communications.

## B. ACTUAL RESULTS

As I mentioned above, the actual test results were not quite what I had expected. From the data I collected, there are several conclusions that I can reliably draw.

First, file transfers between the servers is the slowest on Ethernet and there may be more than one cause. One possible cause is the use of shared memory resources between the processors which could cause delays while waiting for the other processor to complete a memory access cycle. A related cause may be that the multiprocessing operating system is not as efficient as the single processor operating system.

Second, there was a modest performance improvement (approximately 12%) on FDDI over Ethernet on the normal load tests. Table 32 shows the composite improvement results for the normal load tests. I believe this to be as a result of the processing overhead associated with forming the TCP/IP and Ethernet or FDDI packets. H. AlKhatib provides an in depth review of high speed protocol overhead in his lecture notes from the High Performance Local Area Networks tutorial at Compcon Spring '92 [Ref. 39: p. 45].

**Table 32: FDDI IMPROVEMENT OVER ETHERNET (NORMAL LOAD)**

| File size | Percentage Improvement |
|---|---|
| Large file (1155959 bytes) | 12.898 |
| Medium file (58080 bytes) | 12.486 |
| Small file (11 bytes) | 3.6602 |

Table 33 shows the composite improvement results for the no load tests. The improvement is greater here than for the normal load tests because of the dynamic bandwidth allocation scheme associated with FDDI. Notice that the small file did not show much improvement. This is due to the higher percentage of overhead associated with this transfer. If our normal Ethernet loading was greater than G=1 (see Figure 2.3), *i.e.* our network was operating in the saturation region, our normal load improvements would have been similar to those found during the no load tests. FDDI provides the best relative improvement when it replaces an Ethernet network which is routinely saturated and users experience long delays in delivering packets. Recall that throughput on Ethernet can be driven to zero if the offered load (G) gets too high.

**Table 33: FDDI IMPROVEMENT OVER ETHERNET (NO LOAD)**

| File size | Percentage Improvement |
|---|---|
| Large file (1155959 bytes) | 29.215 |
| Medium file (58080 bytes) | 13.682 |
| Small file (11 bytes) | 6.7548 |

When CAPS is being used to develop prototype systems, users can expect at least 12 percent network performance improvement. If several users are actively involved in prototyping work, that improvement will be significantly higher.

## C. COMPARISON

As discussed in Chapter V, Section C, the reason that the theoretical and actual times were very different is because of the processing overhead that is not included in any of the models. The TRT that I assumed to do the calculations was not necessarily the same as what the actual FDDI network uses. The other major unknown is the effect of the concentrator on the network performance. That was not included in any of the models, either. Theoretically, the concentrator should not impose any latency penalties on the network. In reality, it may impose a delay to check for certain network operations, but the delay should be small.

In order to accurately model the true behavior of either Ethernet or FDDI, we would also have to include all the processor tasks. As long as the environment is very tightly specified, it would be possible to calculate the number of machine cycles associated with each process and when processes would be interrupted to perform other tasks for the operating system.

## D. TOPICS FOR FURTHER STUDY

Several topics for further study are related to improving the performance of the software portion of the protocol stacks in one way or another.

For FDDI, one option is currently under study: XTP. Another lightweight protocol under development is TTP which uses a different fiber ring architecture to provide data transfer rates in the Gigabit per second range [Ref. 23: pp. 247–254]. In order to evaluate performance improvements, these protocols would have to be implemented and then integrated with the hardware that is available on CAPSnet. In both cases, these protocols are not compatible with TCP/IP. That would need to be addressed in any study or implementation that was proposed.

Another option would be to investigate ways to improve the TCP/IP suite by decreasing the overhead. One possible implementation would be to use compressed headers the way that CSLIP (Compressed Serial Line Interface Protocol) does for a serial communication interface.

Another possible study area would be performance improvements related to putting the protocol stack in hardware and dedicating an independent (simple but fast) processor to that task. Further, if one processor was dedicated to TCP (or UDP) and another to IP, the only time that they would not run in parallel would be during the first and last datagram processed. Dedicated I/O processors is not a new idea (the

Coast Guard standard workstation used communication input output processors (COMMIOPs) in the early 1980's to improve network performance) and is gaining popularity in consumer grade computers (Apple uses dedicated I/O processors for serial and Ethernet connections in their Quadra line of personal computers).

Finally, an area related to all of these topics would be a way to find out how long each step of the process really takes. Being able to timestamp the major steps would require changing the source code for the UNIX kernel and then recompiling and linking. One major drawback with this approach is that the call to the `gettimeofday()` routine would have to be as unobtrusive as possible so that it would not adversely affect the performance of the stacks. Unfortunately a subroutine call takes much longer than simple memory reads and writes or assignments.

As software engineers provide more powerful applications and users demand higher performance, the demand for improved network performance will continue to accelerate for the foreseeable future.

# APPENDIX A

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

main()
{
    long iteration_array[1000];
    int index, a;
    struct timeval timeslice;
    struct timezone timechunk;

    for( index = 0; index < 1000; index +=1 )

        {
        a = gettimeofday(&timeslice,timechunk);
        iteration_array[index] = timeslice.tv_usec;
        }

    for( index = 0; index < 1000; index +=1 )
        printf("%d\n", iteration_array[index]);

    exit(0),

}
```

# APPENDIX B

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

main()
{

    long elapsed_sec;            /* Seconds variable */
    long elapsed_usec;           /* Microseconds variable */
    float total_time;
    float part_usec;
    int loop_counter;
    int a,b;                     /* Subroutine result vars*/


            /* Variable structure defns */

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

            /* Set up outer loop to execute transfers 50 */
            /* times. */

    for(loop_counter=1; loop_counter<=500; 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 sun52:/tmp/bob.hqx sun53:/tmp/hammar");


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

    b = gettimeofday(&timedone,zonedone);
    if (a != 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 rolling 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 ("The time was %f\n",total_time);

        }       /* This is the end of the control loop. */

        exit(0);

    }
```

# APPENDIX C

| Decimal | Code Group | Symbol | Assignment | |
|---|---|---|---|---|
| Line State Symbols | | | | |
| 00 | 00000 | Q | QUIET | |
| 31 | 11111 | I | IDLE | |
| 04 | 00100 | H | HALT | |
| Starting Delimiter | | | | |
| 24 | 11000 | J | 1st of sequential SD pair | |
| 17 | 10001 | K | 2nd of sequential SD pair | |
| Data Symbols | | | | |
| | | | Hex | Binary |
| 30 | 11110 | 0 | 0 | 0000 |
| 09 | 01001 | 1 | 1 | 0001 |
| 20 | 10100 | 2 | 2 | 0010 |
| 21 | 10101 | 3 | 3 | 0011 |
| 10 | 01010 | 4 | 4 | 0100 |
| 11 | 01011 | 5 | 5 | 0101 |
| 14 | 01110 | 6 | 6 | 0110 |
| 15 | 01111 | 7 | 7 | 0111 |
| 18 | 10010 | 8 | 8 | 1000 |
| 19 | 10011 | 9 | 9 | 1001 |
| 22 | 10110 | A | A | 1010 |
| 23 | 10111 | B | B | 1011 |
| 26 | 11010 | C | C | 1100 |
| 27 | 11011 | D | D | 1101 |
| 28 | 11100 | E | E | 1110 |
| 29 | 11101 | F | F | 1111 |
| Ending Delimiter | | | | |
| 13 | 01101 | T | Used to terminate the data stream | |
| Control indicators | | | | |
| 07 | 00111 | R | Denoting Logical ZERO (Reset) | |
| 25 | 11001 | S | Denoting Logical ONE (Set) | |
| Invalid Code Assignments | | | | |
| 01 | 00001 | V or H | These code patters shall not be transmitted because they violate consecutive code-bit zeros or duty cycle requirements. Codes 01, 02, 08 and 16 shall however be interpreted as Halt when received. | |
| 02 | 00010 | V or H | | |
| 03 | 00011 | V | | |
| 05 | 00101 | V | | |
| 06 | 00110 | V | | |
| 08 | 01000 | V or H | | |
| 12 | 01100 | V | | |
| 16 | 10000 | V or H | | |

# APPENDIX D

## Large Ethernet LAN simulation report.

```
CACI LANNET II.5  RELEASE 3.00    08/25/1992      12:04:44

              Ethernet LAN with two servers and five workstations

                      COLLISION LAN UTILIZATION STATISTICS

                         FROM    0.  TO     2. SECONDS

                      (ALL TIMES REPORTED IN MICROSECONDS)


LAN NAME                     ETHERNET


COLLISION EPISODES               0

COLLIDED TRANSFERS               0
AVG TO RESOLVE                   0.
MAX TO RESOLVE                   0

DEFERRALS                        0
AVG DEFERRAL DELAY               0.
MAX DEFERRAL DELAY               0.
STD DEV DEFERRAL DELAY           0.

AVG DEFERRAL QUEUE               0.
MAX QUEUE SIZE                   0.
STD DEV QUEUE SIZE               0.

MULTIPLE COLLISIONS              0
AVG MULT COLLISIONS              0.
MAX MULT COLLISIONS              0

SUCCESSFUL TRANSFERS             1
AVG USAGE TIME              988596.000
MAX USAGE TIME              988596.000
STD DEV USAGE TIME               0.

PER CENT OF TIME BUSY           49.430
```

COMPLETED ACTIVITY STATISTICS

FROM 0. TO 2. SECONDS

(ALL TIMES REPORTED IN MICROSECONDS)

| ACTIVITY NAME | RECEIVE | SEND A FILE |
|---|---|---|
| HOST STATION | SUNS5 | SUN51 |
| COMPLETED EXECUTIONS | 1 | 1 |
| AVG EXECUTION TIME | 2948.883 | 991544.883 |
| MAX EXECUTION TIME | 2948.883 | 991544.883 |
| MIN EXECUTION TIME | 2948.883 | 991544.883 |
| STD DEV EXECUTION TIME | 0. | 0. |

STATION UTILIZATION STATISTICS

FROM 0. TO 2. SECONDS

(ALL TIMES REPORTED IN MICROSECONDS)

| STATION NAME | SUNS5 | SUNS7 | SUN51 |
|---|---|---|---|
| LAN REQUESTS GRANTED | 0 | 0 | 1 |
| AVERAGE WAIT TIME | 0. | 0. | 0. |
| MAXIMUM WAIT TIME | 0. | 0. | 0. |
| STD DEV WAIT TIME | 0. | 0. | 0. |
| DISK REQUESTS GRANTED | 1 | 0 | 1 |
| AVERAGE BITS USED | 10169444.312 | 0. | 9712400.000 |
| MAXIMUM BITS USED | 10621483. | 0. | 9712400. |
| STD DEV BITS USED | 454534.609 | 0. | 0. |
| STATION UTILIZATION | .147 | 0. | 49.577 |

| STATION NAME | SUN52 | SUN53 | SUN54 |
|---|---|---|---|
| LAN REQUESTS GRANTED | 0 | 0 | 0 |
| AVERAGE WAIT TIME | 0. | 0. | 0. |
| MAXIMUM WAIT TIME | 0. | 0. | 0. |
| STD DEV WAIT TIME | 0. | 0. | 0. |
| DISK REQUESTS GRANTED | 0 | 0 | 0 |
| AVERAGE BITS USED | 0. | 0. | 0. |
| MAXIMUM BITS USED | 0. | 0. | 0. |
| STD DEV BITS USED | 0. | 0. | 0. |
| STATION UTILIZATION | 0. | 0. | 0. |

```
                         RECEIVED MESSAGE REPORT
                       FROM    0.  TO      2. SECONDS

RECEIVER    COUNT        MESSAGE NAME
SUNS5
                 1        LARGE

                         MESSAGE DELIVERY REPORT
                       FROM    0.  TO      2. SECONDS
                    (ALL TIMES REPORTED IN MICROSECONDS)


MESSAGE NAME                    LARGE


SOURCE STATION                  SUN51


DESTINATION STATION             SUNS5


NUMBER SENT                       1
AVG DELIVERY TIME             988596.000
MAX DELIVERY TIME             988596.000
MIN DELIVERY TIME             988596.000
STD DEV DELIVERY TIME             0.
```

## Medium Ethernet LAN report

```
CACI LANNET II.5  RELEASE 3.00    08/25/1992       12:08:18

              Ethernet LAN with two servers and five workstations

                      COLLISION LAN UTILIZATION STATISTICS

                           FROM    0.   TO    .5 SECONDS

                      (ALL TIMES REPORTED IN MICROSECONDS)


LAN NAME                      ETHERNET


COLLISION EPISODES                0

COLLIDED TRANSFERS                0
AVG TO RESOLVE                    0.
MAX TO RESOLVE                    0

DEFERRALS                         0
AVG DEFERRAL DELAY                0.
MAX DEFERRAL DELAY                0.
STD DEV DEFERRAL DELAY            0.

AVG DEFERRAL QUEUE                0.
MAX QUEUE SIZE                    0.
STD DEV QUEUE SIZE                0.

MULTIPLE COLLISIONS               0
AVG MULT COLLISIONS               0.
MAX MULT COLLISIONS               0

SUCCESSFUL TRANSFERS              1
AVG USAGE TIME                49678.300
MAX USAGE TIME                49678.300
STD DEV USAGE TIME                0.

PER CENT OF TIME BUSY             9.936
```

```
                    COMPLETED ACTIVITY STATISTICS

                      FROM    0.  TO    .5 SECONDS

                 (ALL TIMES REPORTED IN MICROSECONDS)


ACTIVITY NAME                  RECEIVE       SEND A FILE


HOST STATION                    SUN55           SUN51


COMPLETED EXECUTIONS              1               1

AVG EXECUTION TIME             982.961        50661.261
MAX EXECUTION TIME             982.961        50661.261
MIN EXECUTION TIME             982.961        50661.261
STD DEV EXECUTION TIME           0.              0.

                    STATION UTILIZATION STATISTICS

                      FROM    0.  TO    .5 SECONDS

                 (ALL TIMES REPORTED IN MICROSECONDS)

STATION NAME                    SUN55           SUN57           SUN51



LAN REQUESTS GRANTED              0               0               1
AVERAGE WAIT TIME                 0.              0.              0.
MAXIMUM WAIT TIME                 0.              0.              0.
STD DEV WAIT TIME                 0.              0.              0.

DISK REQUESTS GRANTED             1               0               1

AVERAGE BITS USED           9753358.197           0.         9712400.000
MAXIMUM BITS USED           9758076.              0.         9712400.
STD DEV BITS USED             13900.817           0.              0.

STATION UTILIZATION                .197           0.            10.132


STATION NAME                    SUN52           SUN53           SUN54



LAN REQUESTS GRANTED              0               0               0
AVERAGE WAIT TIME                 0.              0.              0.
MAXIMUM WAIT TIME                 0.              0.              0.
STD DEV WAIT TIME                 0.              0.              0.

DISK REQUESTS GRANTED             0               0               0

AVERAGE BITS USED                 0.              0.              0.
MAXIMUM BITS USED                 0.              0.              0.
STD DEV BITS USED                 0.              0.              0.

STATION UTILIZATION               0.              0.              0.
```

```
                    STATION UTILIZATION STATISTICS

                      FROM    0.   TO    .5 SECONDS

                 (ALL TIMES REPORTED IN MICROSECONDS)

STATION NAME                    SUN55



LAN REQUESTS GRANTED              0
AVERAGE WAIT TIME                 0.
MAXIMUM WAIT TIME                 0.
STD DEV WAIT TIME                 0.

DISK REQUESTS GRANTED             0

AVERAGE BITS USED                 0.
MAXIMUM BITS USED                 0.
STD DEV BITS USED                 0.

STATION UTILIZATION               0.


                      RECEIVED MESSAGE REPORT

                      FROM    0.   TO    .5 SECONDS

RECEIVER    COUNT         MESSAGE NAME

SUN55
               1          MEDIUM

                      MESSAGE DELIVERY REPORT

                      FROM    0.   TO    .5 SECONDS

                 (ALL TIMES REPORTED IN MICROSECONDS)


MESSAGE NAME                    MEDIUM


SOURCE STATION                  SUN51


DESTINATION STATION             SUN55


NUMBER SENT                       1
AVG DELIVERY TIME               49678.300
MAX DELIVERY TIME               49678.301
MIN DELIVERY TIME               49678.301
STD DEV DELIVERY TIME             0.
```

## Small Ethernet LAN report

```
CACI LANNET II.5  RELEASE 3.00    08/25/1992       12:11:13
                Ethernet LAN with two servers and five workstations

                         COLLISION LAN UTILIZATION STATISTICS
                              FROM   0.   TO   .1 SECONDS
                         (ALL TIMES REPORTED IN MICROSECONDS)


LAN NAME                         ETHERNET


COLLISION EPISODES                  0

COLLIDED TRANSFERS                  0
AVG TO RESOLVE                      0.
MAX TO RESOLVE                      0

DEFERRALS                           0
AVG DEFERRAL DELAY                  0.
MAX DEFERRAL DELAY                  0.
STD DEV DEFERRAL DELAY              0.

AVG DEFERRAL QUEUE                  0.
MAX QUEUE SIZE                      0.
STD DEV QUEUE SIZE                  0.

MULTIPLE COLLISIONS                 0
AVG MULT COLLISIONS                 0.
MAX MULT COLLISIONS                 0

SUCCESSFUL TRANSFERS                1
AVG USAGE TIME                     85.600
MAX USAGE TIME                     85.600
STD DEV USAGE TIME                  0.

PER CENT OF TIME BUSY                .086
```

```
                         COMPLETED ACTIVITY STATISTICS

                          FROM    0.  TO    .1 SECONDS

                     (ALL TIMES REPORTED IN MICROSECONDS)


ACTIVITY NAME                   RECEIVE         SEND A FILE


HOST STATION                    SUN55           SUN51


COMPLETED EXECUTIONS              1               1

AVG EXECUTION TIME             982.961         1068.561
MAX EXECUTION TIME             982.961         1068.561
MIN EXECUTION TIME             982.961         1068.561
STD DEV EXECUTION TIME           0.              0.

                         STATION UTILIZATION STATISTICS

                          FROM    0.  TO    .1 SECONDS

                     (ALL TIMES REPORTED IN MICROSECONDS)

STATION NAME                    SUN55           SUN57           SUN51



LAN REQUESTS GRANTED              0               0               1
AVERAGE WAIT TIME                 0.              0.              0.
MAXIMUM WAIT TIME                 0.              0.              0.
STD DEV WAIT TIME                 0.              0.              0.

DISK REQUESTS GRANTED             1               0               1

AVERAGE BITS USED            9713459.803          0.          9712400.000
MAXIMUM BITS USED            9713482.             0.          9712400.
STD DEV BITS USED                153.378          0.              0.

STATION UTILIZATION                 .983          0.              1.069


STATION NAME                    SUN52           SUN53           SUN54



LAN REQUESTS GRANTED              0               0               0
AVERAGE WAIT TIME                 0.              0.              0.
MAXIMUM WAIT TIME                 0.              0               0.
STD DEV WAIT TIME                 0.              0.              0.

DISK REQUESTS GRANTED             0               0               0

AVERAGE BITS USED                 0.              0.              0.
MAXIMUM BITS USED                 0.              0.              0.
STD DEV BITS USED                 0.              0.              0.

STATION UTILIZATION               0.              0.              0.
```

```
                         STATION UTILIZATION STATISTICS

                          FROM    0.  TO    .1 SECONDS

                      (ALL TIMES REPORTED IN MICROSECONDS)

STATION NAME                     SUN55



LAN REQUESTS GRANTED               0
AVERAGE WAIT TIME                  0.
MAXIMUM WAIT TIME                  0.
STD DEV WAIT TIME                  0.

DISK REQUESTS GRANTED              0

AVERAGE BITS USED                  0.
MAXIMUM BITS USED                  0.
STD DEV BITS USED                  0.

STATION UTILIZATION                0.


                           RECEIVED MESSAGE REPORT

                           FROM    0.  TO    .1 SECONDS


RECEIVER     COUNT          MESSAGE NAME

SUNS5
                 1          SMALL

                           MESSAGE DELIVERY REPORT

                           FROM    0.  TO    .1 SECONDS

                      (ALL TIMES REPORTED IN MICROSECONDS)


MESSAGE NAME                     SMALL


SOURCE STATION                   SUN51


DESTINATION STATION              SUNS5


NUMBER SENT                        1
AVG DELIVERY TIME                85.600
MAX DELIVERY TIME                85.600
MIN DELIVERY TIME                85.600
STD DEV DELIVERY TIME              0.
```

## Large FDDI LAN report

```
CACI LANNET II.5  RELEASE 3.00    08/25/1992      11:53:04

                FDDI LAN with two servers and five workstations.

                         TOKEN LAN UTILIZATION STATISTICS

                          FROM    0.  TO      2. SECONDS

                       (ALL TIMES REPORTED IN MICROSECONDS)


LAN NAME                        FDDI



LAN REQUESTS GRANTED               1
AVG REQUEST DELAY                  0.
MAX REQUEST DELAY                  0.
STD DEV REQUEST DELAY              0.

COMPLETED TRANSFERS                1
AVG USAGE TIME                 94419.232
MAX USAGE TIME                 94419.232
STD DEV USAGE TIME                 0.

AVG QUEUE SIZE                     0.
MAX QUEUE SIZE                     1.000
STD DEV QUEUE SIZE                 0.

PER CENT OF TIME BUSY              4.721


                          COMPLETED ACTIVITY STATISTICS

                          FROM    0.  TO      2. SECONDS

                       (ALL TIMES REPORTED IN MICROSECONDS)


ACTIVITY NAME                   RECEIVE          SEND FILE


HOST STATION                     SUNS5             SUN51


COMPLETED EXECUTIONS               1                 1

AVG EXECUTION TIME              2948.883          97368.115
MAX EXECUTION TIME              2948.883          97368.115
MIN EXECUTION TIME              2948.883          97368.115
STD DEV EXECUTION TIME             0.                0.
```

```
                    STATION UTILIZATION STATISTICS

                     FROM    0.  TO      2. SECONDS

                  (ALL TIMES REPORTED IN MICROSECONDS)

STATION NAME               SUNS5           SUNS7           SUN51



LAN REQUESTS GRANTED         0               0               1
AVERAGE WAIT TIME            0.              0.              0.
MAXIMUM WAIT TIME            0.              0.              0.
STD DEV WAIT TIME            0.              0.              0.

DISK REQUESTS GRANTED        1               0               1

AVERAGE BITS USED       18779553.724     9712400.000     9712400.000
MAXIMUM BITS USED       19258365.        9712400.        9712400.
STD DEV BITS USED        2083615.954           0.              0.

STATION UTILIZATION             .147           0.              4.868



STATION NAME               SUN52           SUN53           SUN54



LAN REQUESTS GRANTED         0               0               0
AVERAGE WAIT TIME            0.              0.              0.
MAXIMUM WAIT TIME            0.              0.              0.
STD DEV WAIT TIME            0.              0.              0.

DISK REQUESTS GRANTED        0               0               0

AVERAGE BITS USED         9712400.000     9712400.000     9712400.000
MAXIMUM BITS USED         9712400.        9712400.        9712400.
STD DEV BITS USED               0.              0.              0.

STATION UTILIZATION             0.              0.              0.



STATION NAME               SUN55



LAN REQUESTS GRANTED         0
AVERAGE WAIT TIME            0.
MAXIMUM WAIT TIME            0.
STD DEV WAIT TIME            0.

DISK REQUESTS GRANTED        0

AVERAGE BITS USED         9712400.000
MAXIMUM BITS USED         9712400.
STD DEV BITS USED               0.

STATION UTILIZATION             0.
```

```
                         RECEIVED MESSAGE REPORT

                        FROM    0.  TO      2. SECONDS

RECEIVER    COUNT        MESSAGE NAME

SUNS5
                1        LARGE

                         MESSAGE DELIVERY REPORT

                        FROM    0.  TO      2. SECONDS

                   (ALL TIMES REPORTED IN MICROSECONDS)


MESSAGE NAME                   LARGE


SOURCE STATION                 SUNS1


DESTINATION STATION            SUNS5


NUMBER SENT                        1
AVG DELIVERY TIME           94419.232
MAX DELIVERY TIME           94419.234
MIN DELIVERY TIME           94419.234
STD DEV DELIVERY TIME           0.
```

## Medium FDDI LAN report

```
CACI LANNET II.5  RELEASE 3.00    08/25/1992      11:56:31

                  FDDI LAN with two servers and five workstations.

                          TOKEN LAN UTILIZATION STATISTICS

                             FROM    0.   TO    .5 SECONDS

                        (ALL TIMES REPORTED IN MICROSECONDS)


LAN NAME                            FDDI



LAN REQUESTS GRANTED                  1
AVG REQUEST DELAY                     0.
MAX REQUEST DELAY                     0.
STD DEV REQUEST DELAY                 0.

COMPLETED TRANSFERS                   1
AVG USAGE TIME                     4745.384
MAX USAGE TIME                     4745.384
STD DEV USAGE TIME                    0.

AVG QUEUE SIZE                        0.
MAX QUEUE SIZE                        1.000
STD DEV QUEUE SIZE                    0.

PER CENT OF TIME BUSY                  .949


                           COMPLETED ACTIVITY STATISTICS

                             FROM    0.   TO    .5 SECONDS

                        (ALL TIMES REPORTED IN MICROSECONDS)


ACTIVITY NAME                      RECEIVE          SEND FILE


HOST STATION                        SUN55             SUN51


COMPLETED EXECUTIONS                  1                 1

AVG EXECUTION TIME                  982.961          5728.345
MAX EXECUTION TIME                  982.961          5728.345
MIN EXECUTION TIME                  982.961          5728.345
STD DEV EXECUTION TIME                0.                0.
```

```
                         STATION UTILIZATION STATISTICS

                          FROM    0.   TO    .5 SECONDS

                       (ALL TIMES REPORTED IN MICROSECONDS)

STATION NAME                  SUNS5           SUNS7           SUN51



LAN REQUESTS GRANTED            0               0               1
AVERAGE WAIT TIME               0.              0.              0.
MAXIMUM WAIT TIME               0.              0.              0.
STD DEV WAIT TIME               0.              0.              0.

DISK REQUESTS GRANTED           1               0               1

AVERAGE BITS USED        10185589.153     9712400.000     9712400.000
MAXIMUM BITS USED        10192027.        9712400.        9712400.
STD DEV BITS USED           55193.473           0.              0.

STATION UTILIZATION              .197           0.              1.146



STATION NAME                  SUN52           SUN53           SUN54



LAN REQUESTS GRANTED            0               0               0
AVERAGE WAIT TIME               0.              0.              0.
MAXIMUM WAIT TIME               0.              0.              0.
STD DEV WAIT TIME               0.              0.              0.

DISK REQUESTS GRANTED           0               0               0

AVERAGE BITS USED          9712400.000     9712400.000     9712400.000
MAXIMUM BITS USED          9712400.        9712400.        9712400.
STD DEV BITS USED               0.              0.              0.

STATION UTILIZATION             0.              0.              0.



STATION NAME                  SUN55



LAN REQUESTS GRANTED            0
AVERAGE WAIT TIME               0.
MAXIMUM WAIT TIME               0.
STD DEV WAIT TIME               0.

DISK REQUESTS GRANTED           0

AVERAGE BITS USED          9712400.000
MAXIMUM BITS USED          9712400.
STD DEV BITS USED               0.

STATION UTILIZATION             0.
```

```
                          RECEIVED MESSAGE REPORT
                         FROM    0.  TO    .5 SECONDS

RECEIVER    COUNT         MESSAGE NAME
SUNS5
                 1        MEDIUM
                          MESSAGE DELIVERY REPORT
                         FROM    0.  TO    .5 SECONDS
                    (ALL TIMES REPORTED IN MICROSECONDS)


MESSAGE NAME                     MEDIUM


SOURCE STATION                   SUNS1


DESTINATION STATION              SUNS5


NUMBER SENT                         1
AVG DELIVERY TIME                4745.384
MAX DELIVERY TIME                4745.384
MIN DELIVERY TIME                4745.384
STD DEV DELIVERY TIME               0.
```

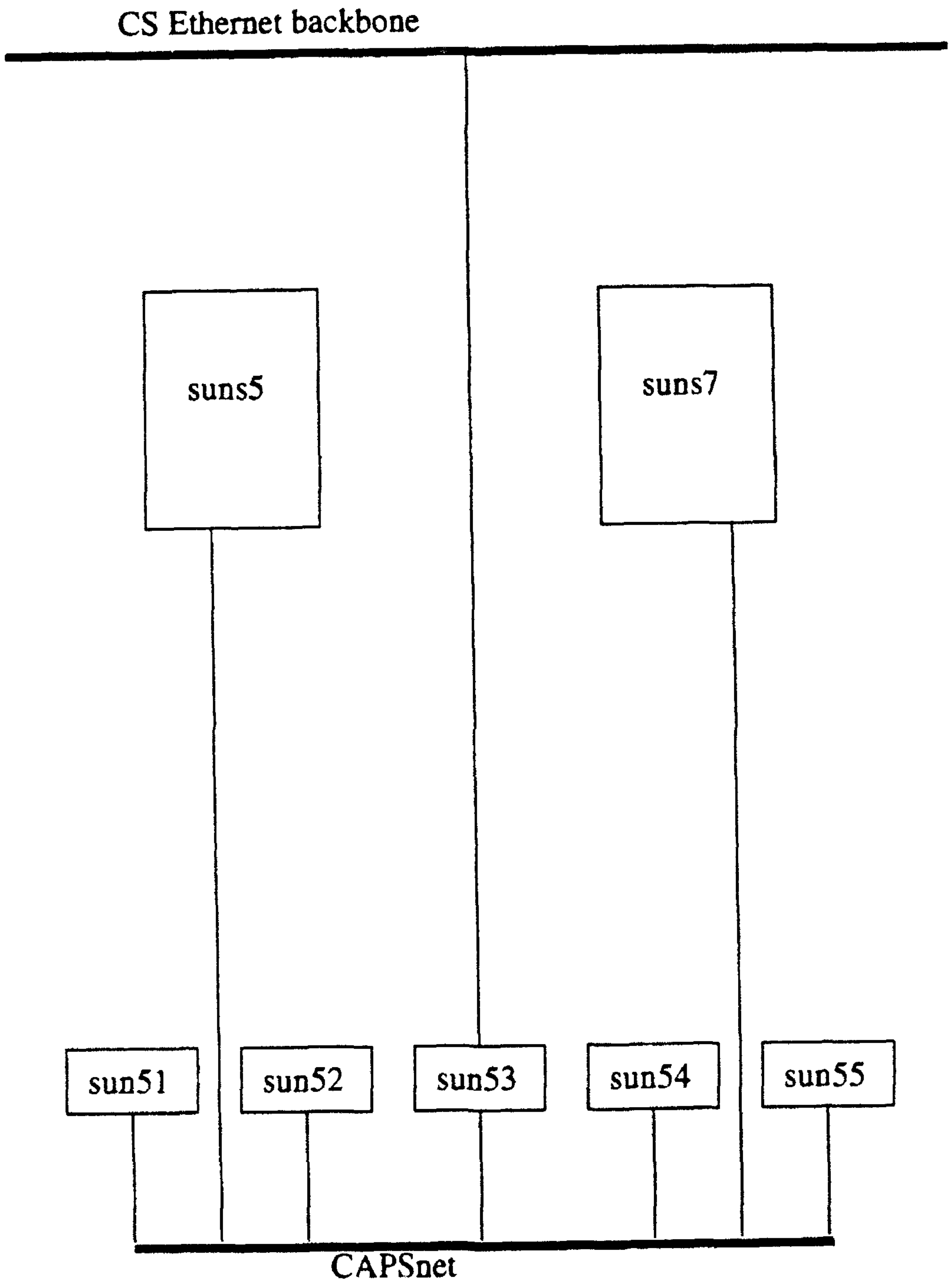
## Small FDDI LAN report

```
CACI LANNET II.5  RELEASE 3.00    08/25/1992       11:59:36

                FDDI LAN with two servers and five workstations.

                        TOKEN LAN UTILIZATION STATISTICS

                           FROM    0.   TO    .1 SECONDS

                       (ALL TIMES REPORTED IN MICROSECONDS)


LAN NAME                       FDDI



LAN REQUESTS GRANTED             1
AVG REQUEST DELAY                0.
MAX REQUEST DELAY                0.
STD DEV REQUEST DELAY            0.

COMPLETED TRANSFERS              1
AVG USAGE TIME                   8.136
MAX USAGE TIME                   8.136
STD DEV USAGE TIME               0.

AVG QUEUE SIZE                   0.
MAX QUEUE SIZE                   1.000
STD DEV QUEUE SIZE               0.

PER CENT OF TIME BUSY             .008


                        COMPLETED ACTIVITY STATISTICS

                           FROM    0.   TO    .1 SECONDS

                       (ALL TIMES REPORTED IN MICROSECONDS)


ACTIVITY NAME                  RECEIVE          SEND FILE


HOST STATION                    SUNS5            SUN51


COMPLETED EXECUTIONS              1                1

AVG EXECUTION TIME             982.961          991.097
MAX EXECUTION TIME             982.961          991.097
MIN EXECUTION TIME             982.961          991.097
STD DEV EXECUTION TIME           0.               0.
```

```
                         STATION UTILIZATION STATISTICS

                          FROM    0.   TO    .1 SECONDS

                        (ALL TIMES REPORTED IN MICROSECONDS)

STATION NAME                 SUNS5           SUNS7           SUN51



LAN REQUESTS GRANTED            0               0               1
AVERAGE WAIT TIME               0.              0.              0.
MAXIMUM WAIT TIME               0.              0.              0.
STD DEV WAIT TIME               0.              0.              0.

DISK REQUESTS GRANTED           1               0               1

AVERAGE BITS USED        9713566.509     9712400.000     9712400.000
MAXIMUM BITS USED        9713590.        9712400.        9712400.
STD DEV BITS USED            165.538           0.              0.

STATION UTILIZATION             .983            0.              .991


STATION NAME                 SUN52           SUN53           SUN54



LAN REQUESTS GRANTED            0               0               0
AVERAGE WAIT TIME               0.              0.              0.
MAXIMUM WAIT TIME               0.              0.              0.
STD DEV WAIT TIME               0.              0.              0.

DISK REQUESTS GRANTED           0               0               0

AVERAGE BITS USED        9712400.000     9712400.000     9712400.000
MAXIMUM BITS USED        9712400.        9712400.        9712400.
STD DEV BITS USED               0.              0.              0.

STATION UTILIZATION             0.              0.              0.


STATION NAME                 SUN55



LAN REQUESTS GRANTED            0
AVERAGE WAIT TIME               0.
MAXIMUM WAIT TIME               0.
STD DEV WAIT TIME               0.

DISK REQUESTS GRANTED           0

AVERAGE BITS USED        9712400.000
MAXIMUM BITS USED        9712400.
STD DEV BITS USED               0.

STATION UTILIZATION             0.
```

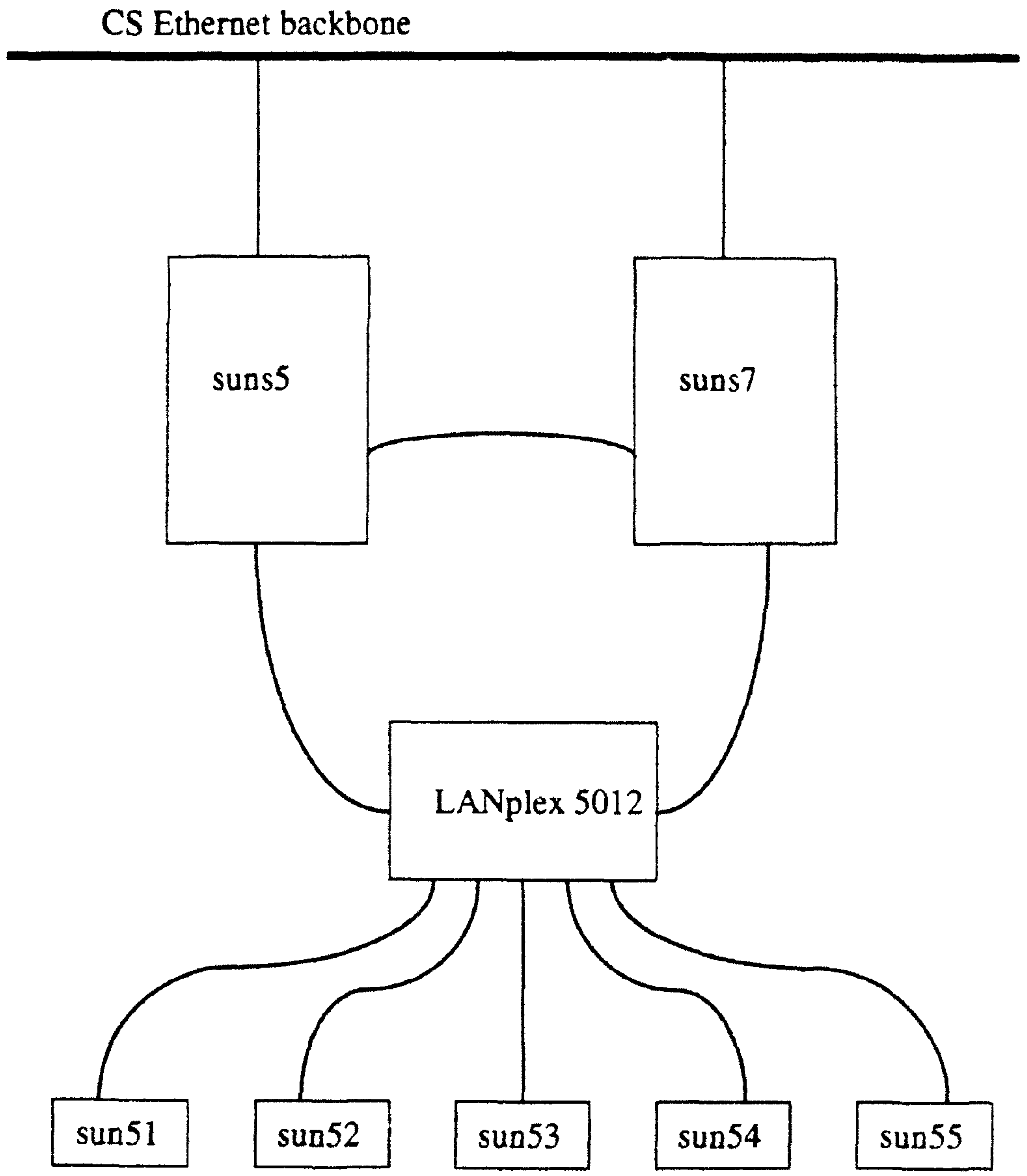
```
                         RECEIVED MESSAGE REPORT

                        FROM    0.   TO    .1 SECONDS

RECEIVER    COUNT       MESSAGE NAME

SUNS5
               1        SMALL

                         MESSAGE DELIVERY REPORT

                        FROM    0.   TO    .1 SECONDS

                   (ALL TIMES REPORTED IN MICROSECONDS)


MESSAGE NAME                  SMALL


SOURCE STATION                SUN51


DESTINATION STATION           SUNS5


NUMBER SENT                     1
AVG DELIVERY TIME               8.136
MAX DELIVERY TIME               8.136
MIN DELIVERY TIME               8.136
STD DEV DELIVERY TIME           0.
```

## APPENDIX E



CAPSnet before the changes. Orignially, CAPSnet was an Ethernet subnet.

## APPENDIX F



CAPSnet will have the topology shown above after FDDI is fully installed. Each single line represents a duplex fiber.

## APPENDIX G

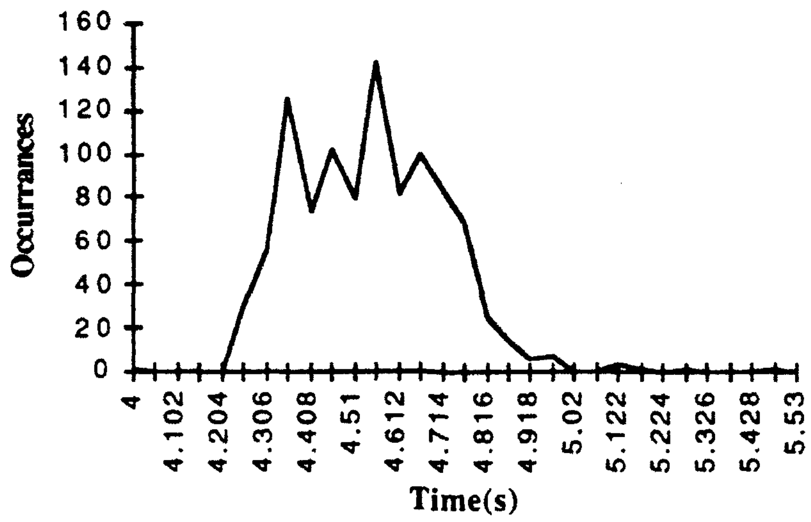


Figure G.1 Test #1, Group #1, large file on Ethernet

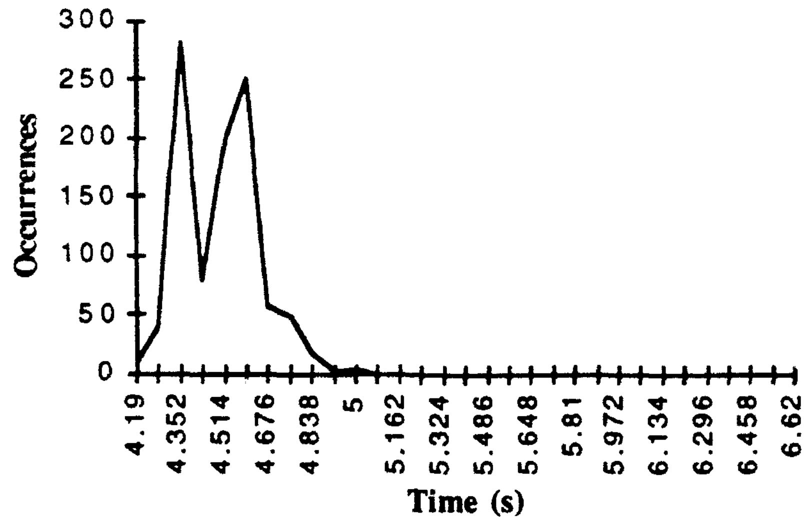


Figure G.2 Test #1, Group #2, large file on Ethernet

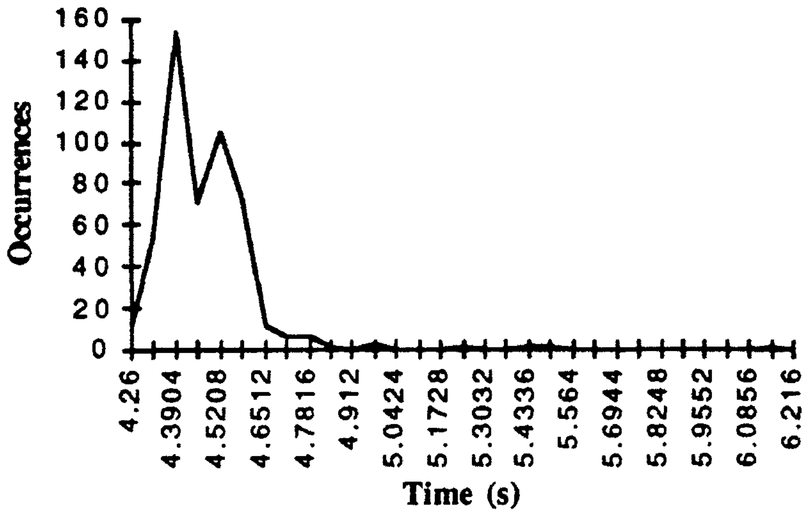


Figure G.3 Test #1, Group #3, large file on Ethernet

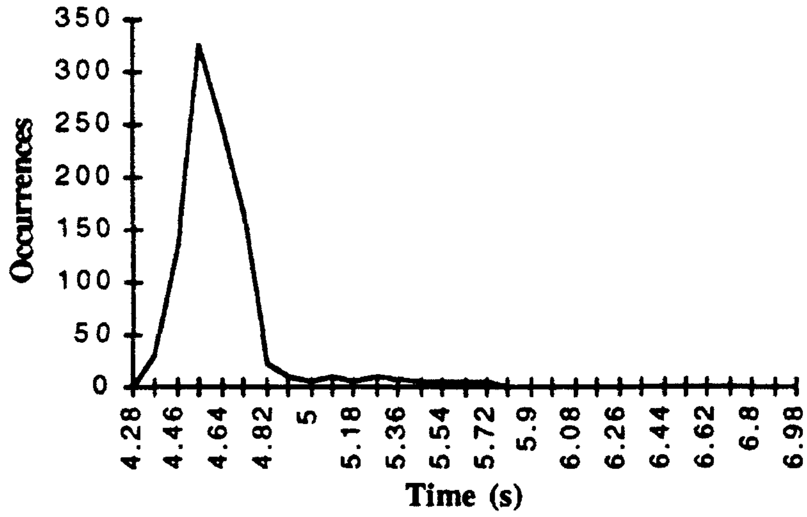


Figure G.4 Test #1, Group #4, large file on Ethernet

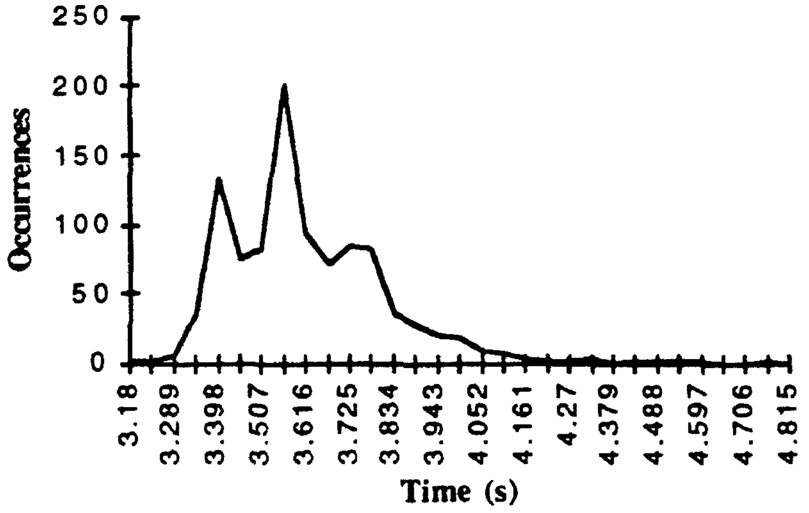


Figure G.5 Test #1, Group #1, medium file on Ethernet

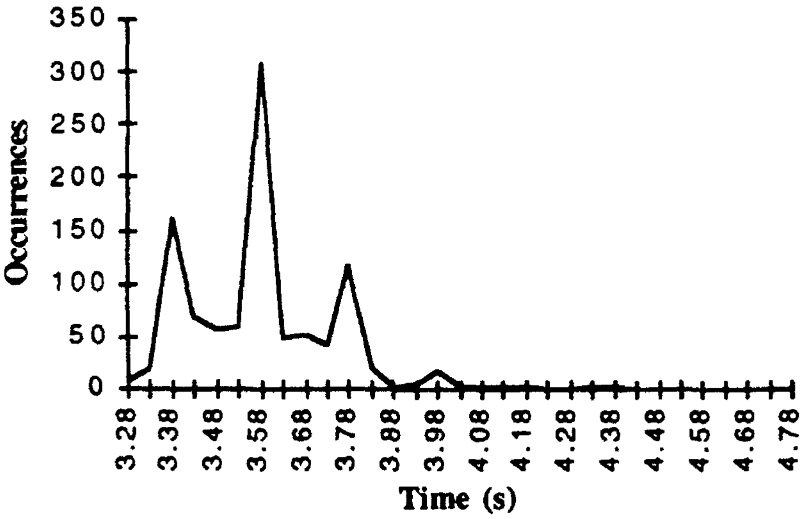


Figure G.6 Test #1, Group #2, medium file on Ethernet

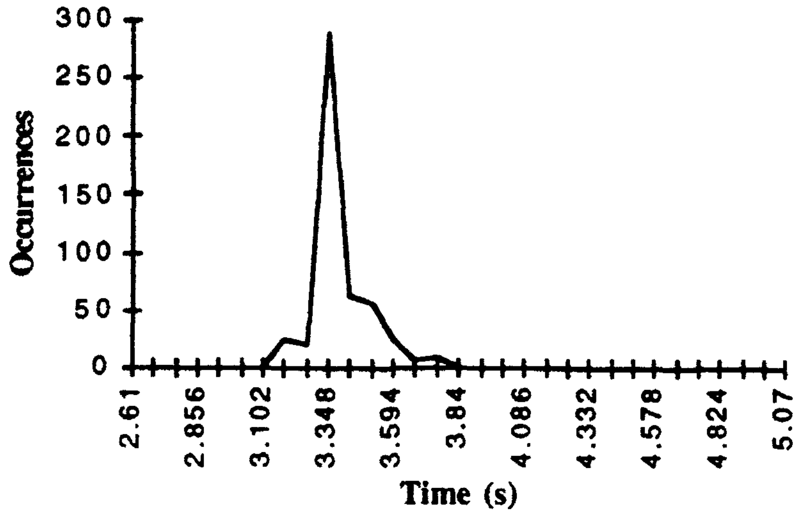


**Figure G.7 Test #1, Group #3, medium file on Ethernet**

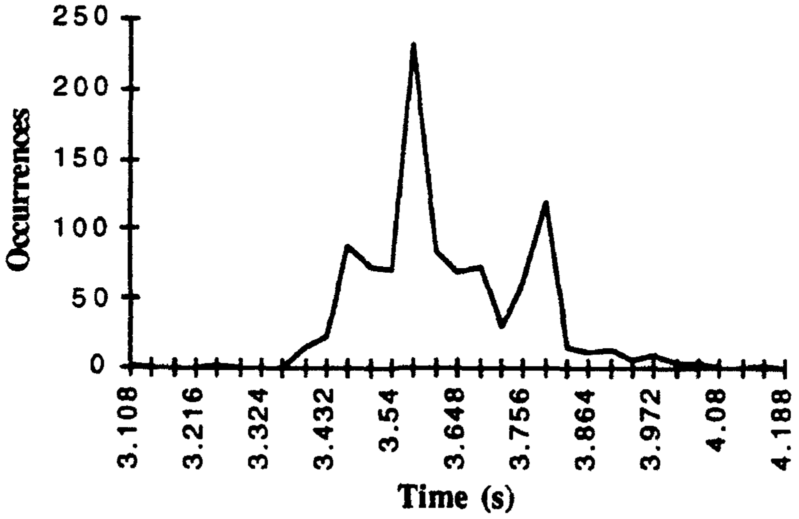


**Figure G.8 Test #1, Group #4, medium file on Ethernet**

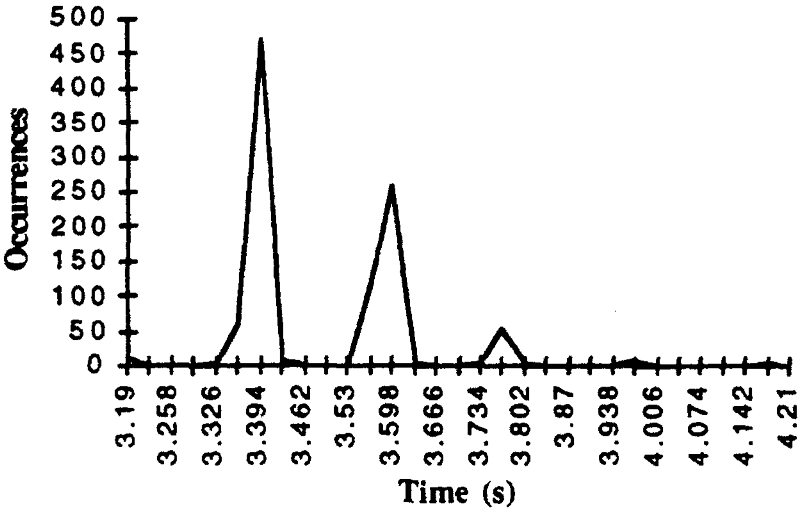


**Figure G.9 Test #1, Group #1, small file on Ethernet**

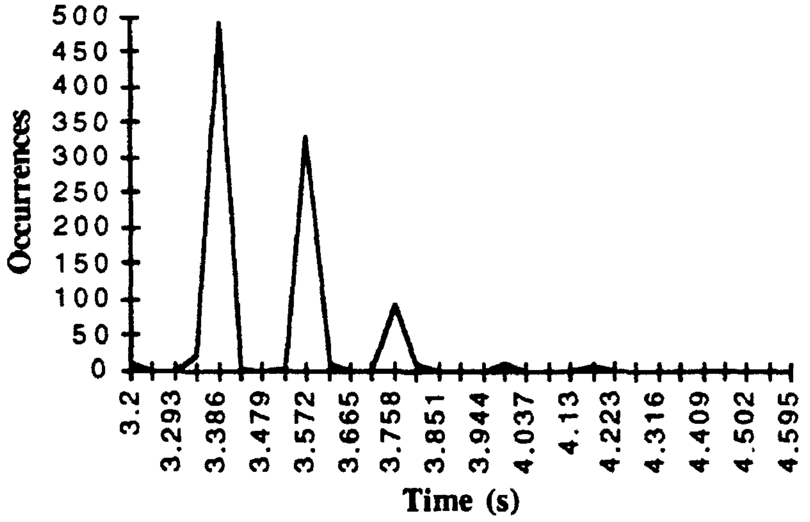


**Figure G.10 Test #1, Group #2, small file on Ethernet**

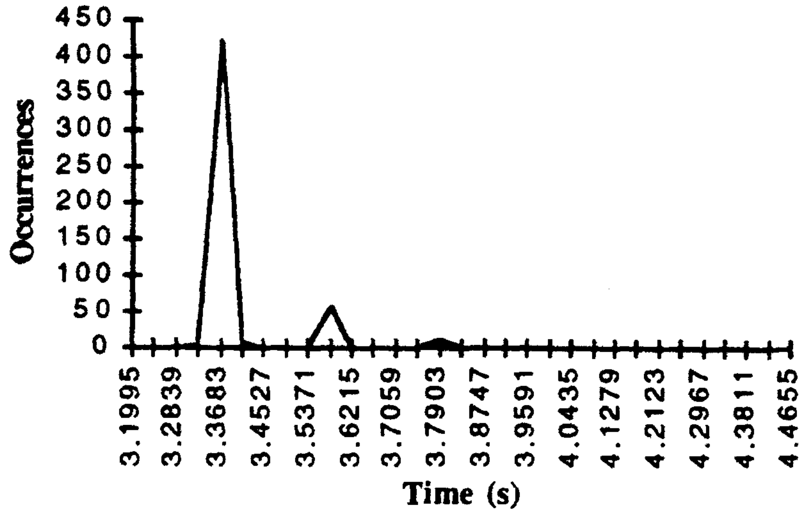


Figure G.11 Test #1, Group #3, small file on Ethernet

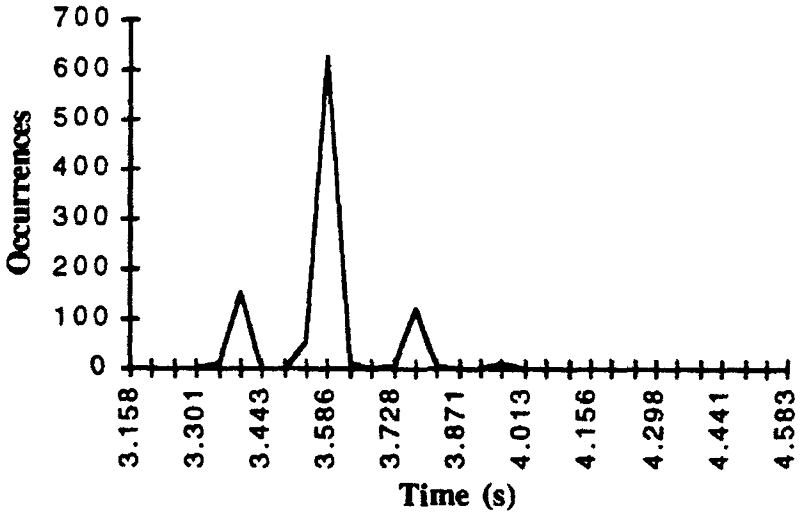


Figure G.12 Test #1, Group #4, small file on Ethernet

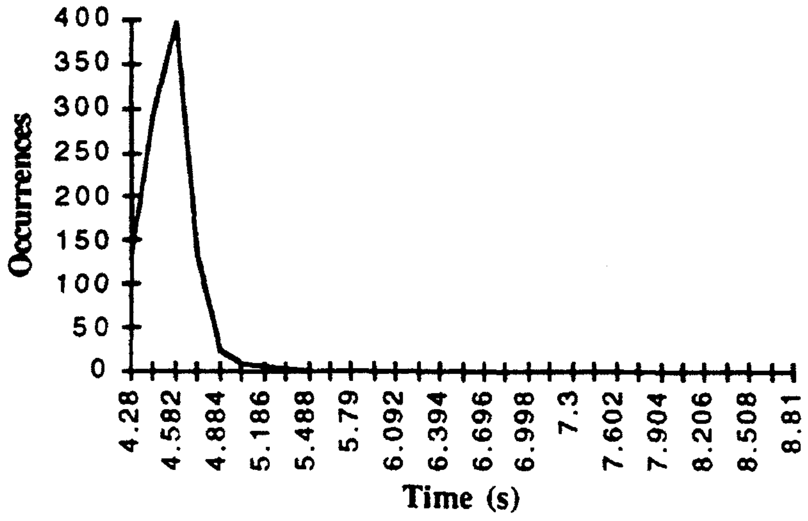


Figure G.13 Test #2, Group #1, large file on Ethernet

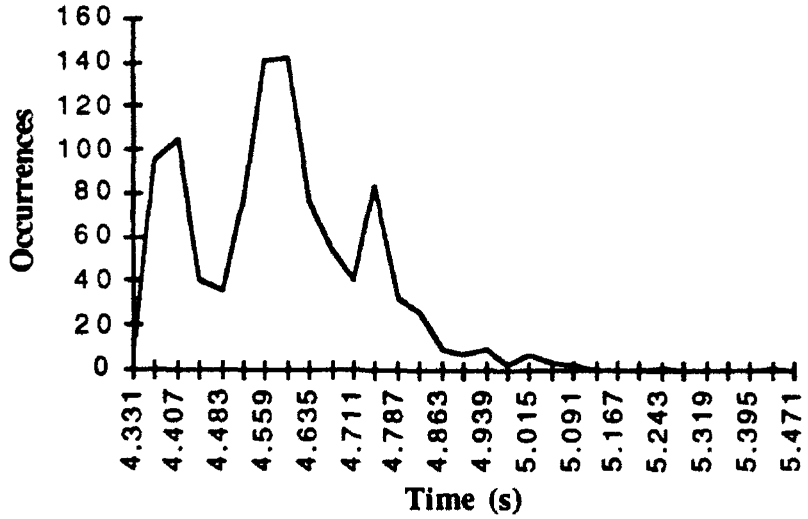


Figure G.14 Test #2, Group #2, large file on Ethernet

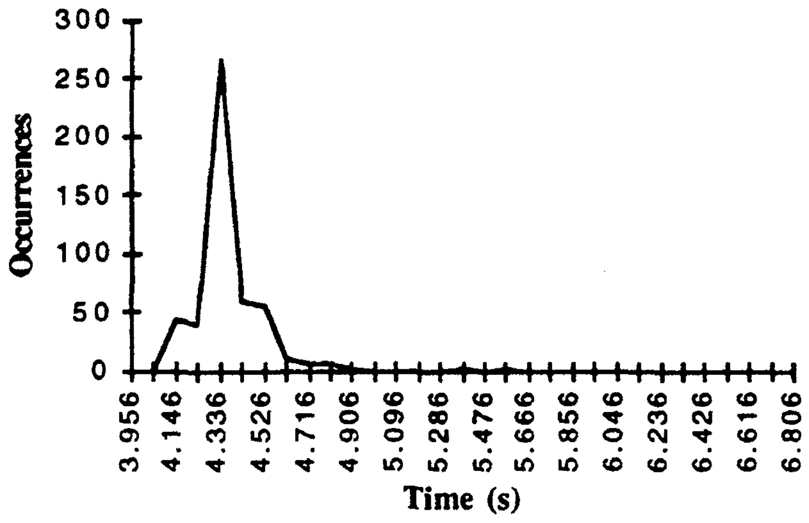


Figure G.15 Test #2, Group #3, large file on Ethernet

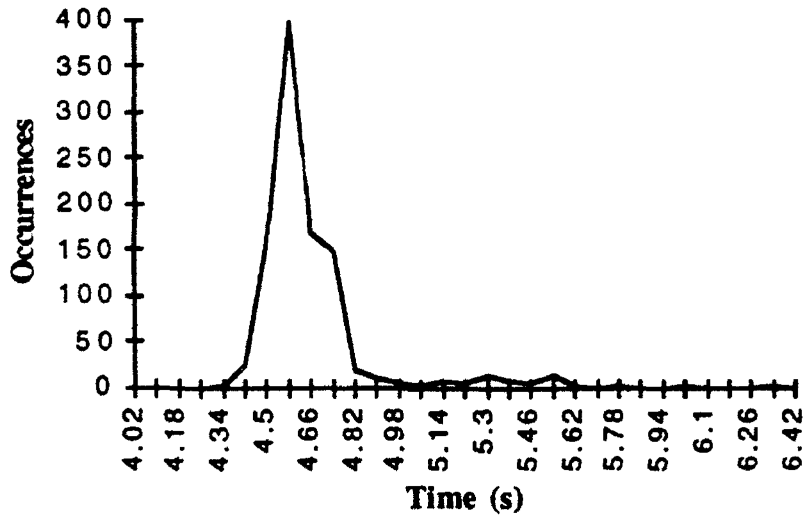


Figure G.16 Test #2, Group #4, large file on Ethernet

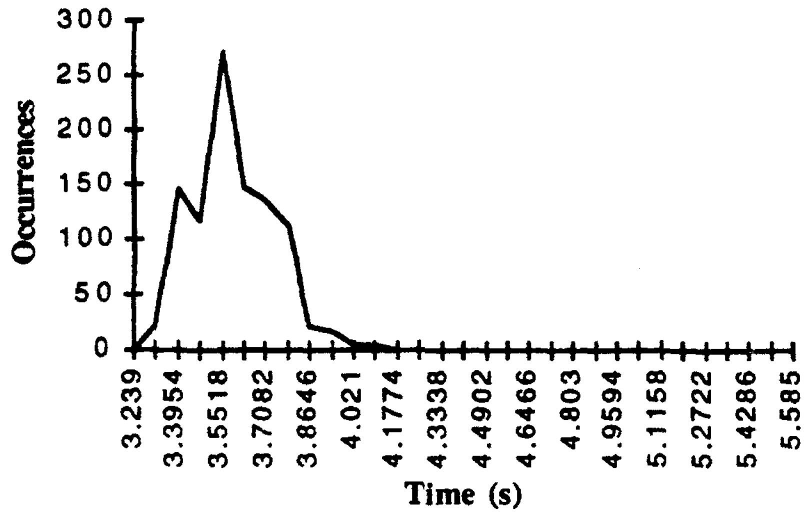


Figure G.17 Test #2, Group #1, medium file on Ethernet

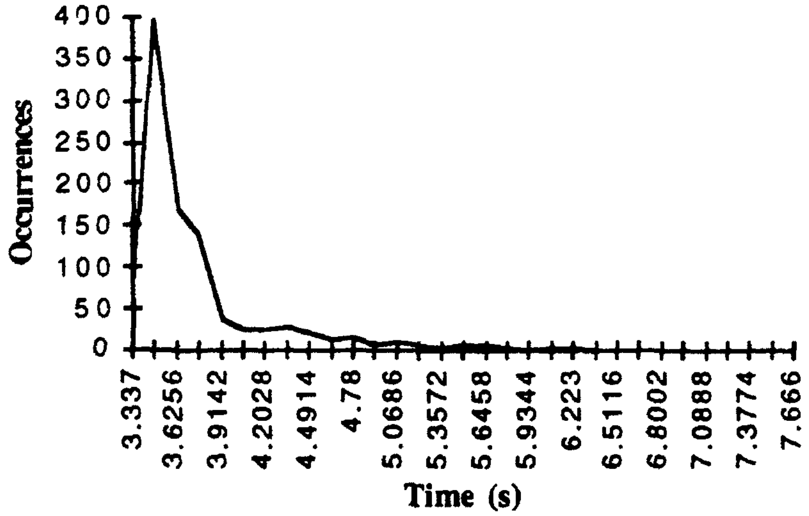


Figure G.18 Test #2, Group #2, medium file on Ethernet

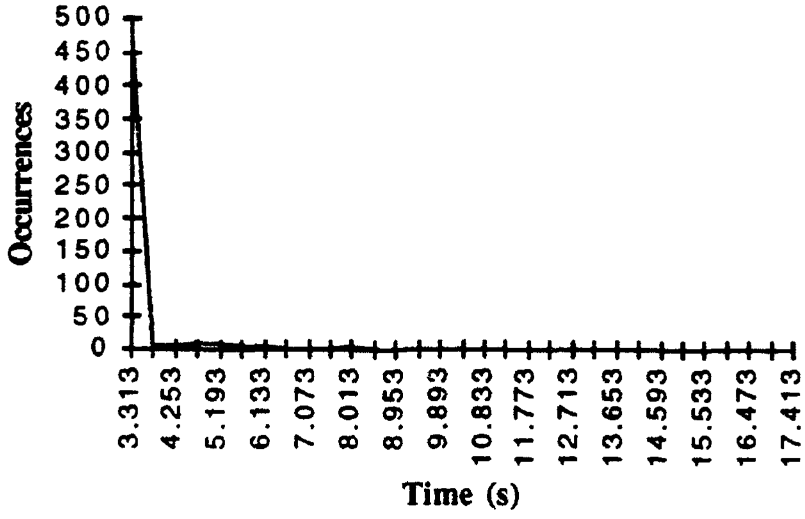


Figure G.19 Test #2, Group #3, medium file on Ethernet

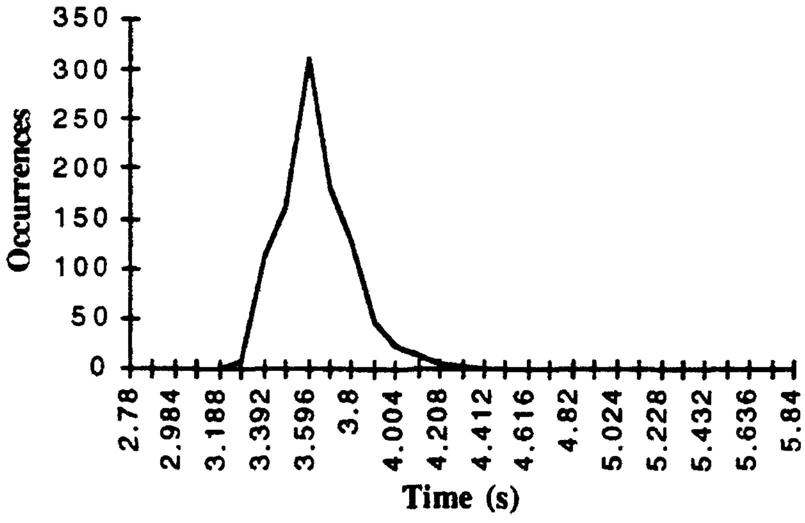


Figure G.20 Test #2, Group #4, medium file on Ethernet

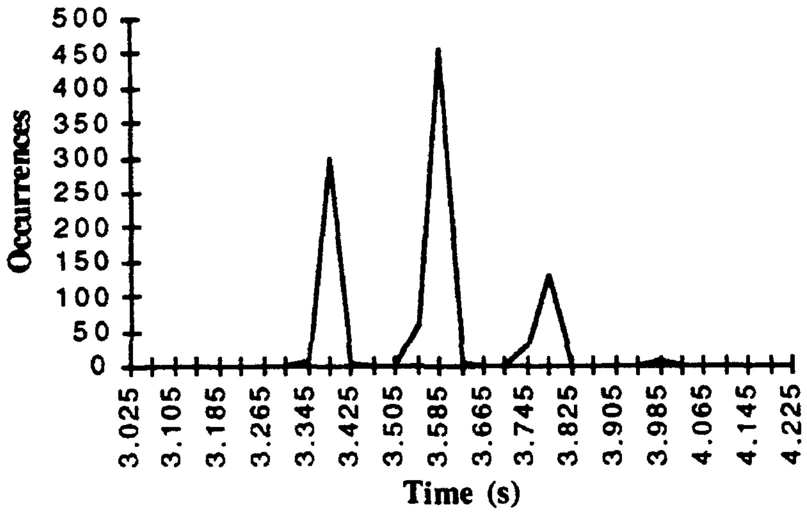


**Figure G.21 Test #2, Group #1, small file on Ethernet**

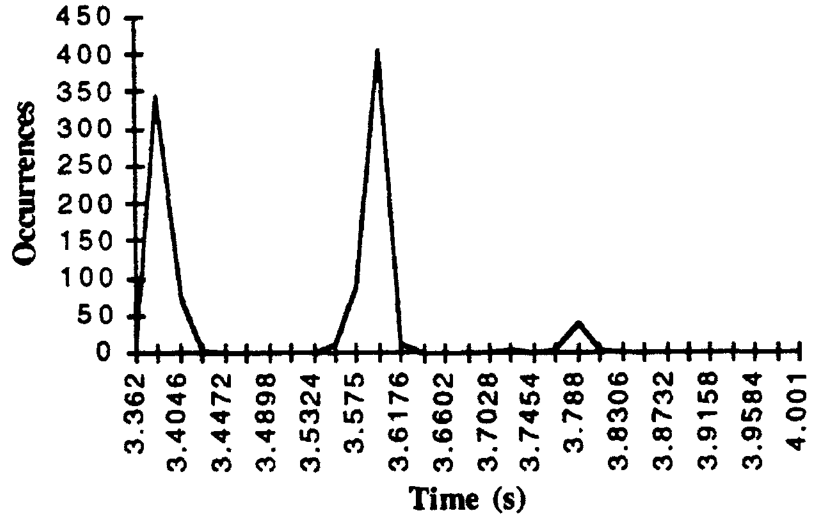


**Figure G.22 Test #2, Group #2, small file on Ethernet**

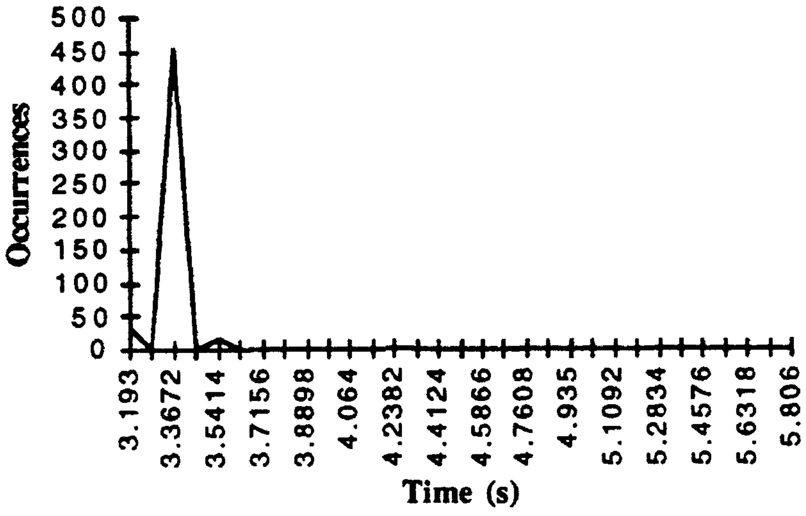


**Figure G.23 Test #2, Group #3, small file on Ethernet**

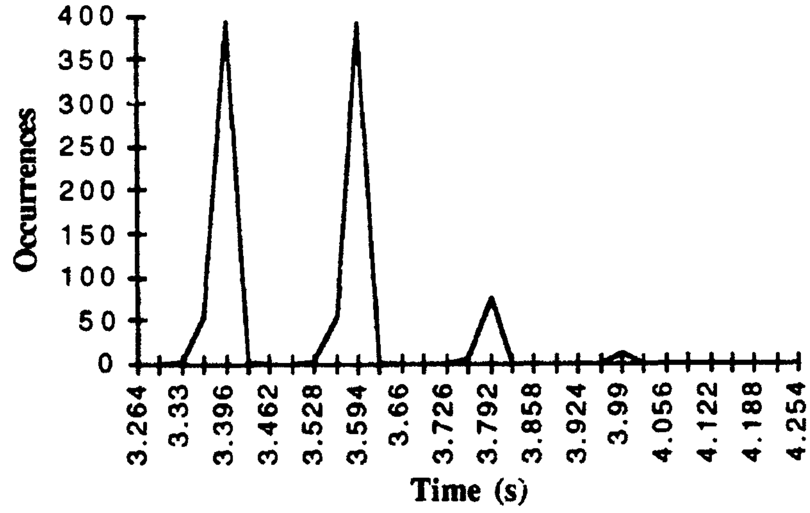


**Figure G.24 Test #2, Group #4, small file on Ethernet**

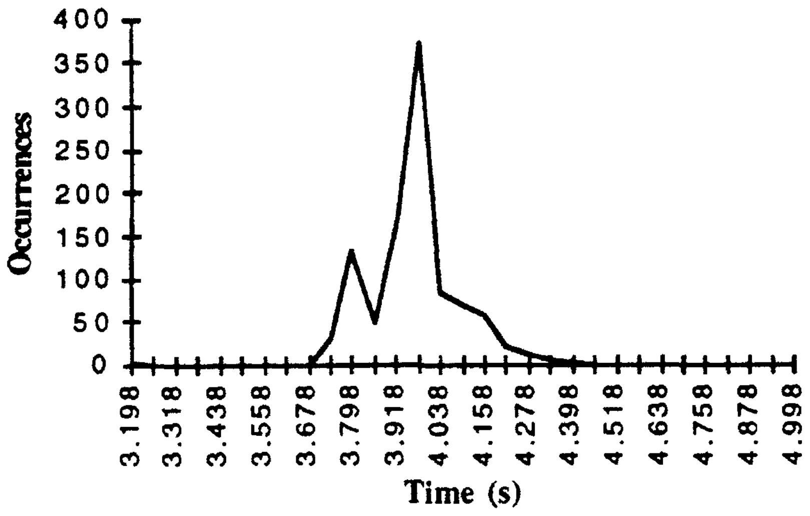


Figure G.25 Test #3, Group #1, large file on FDDI

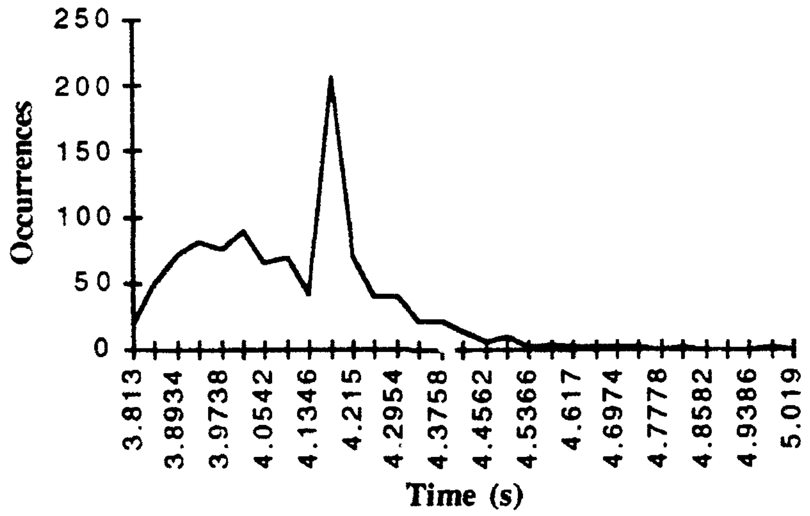


Figure G.26 Test #3, Group #2, large file on FDDI

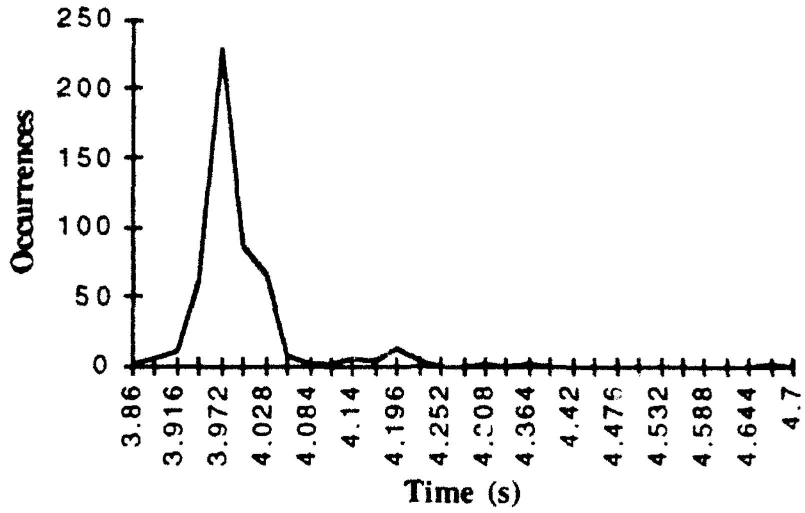


Figure G.27 Test #3, Group #3, large file on FDDI

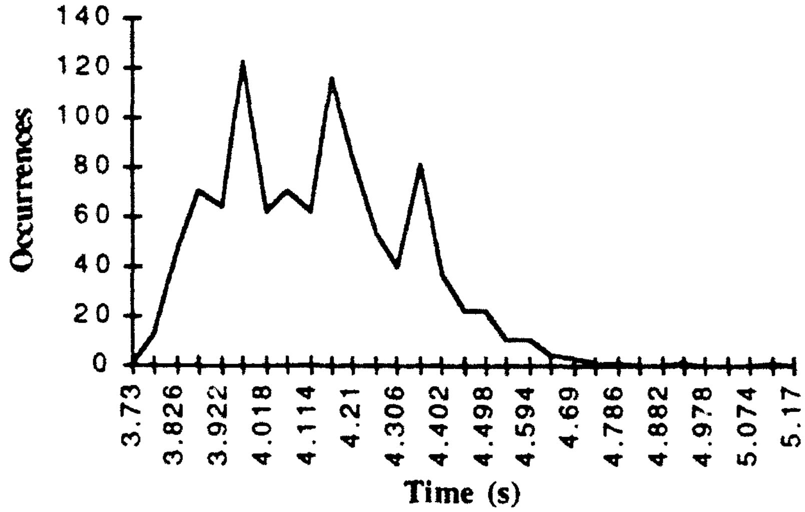


Figure G.28 Test #3, Group #4, large file on FDDI

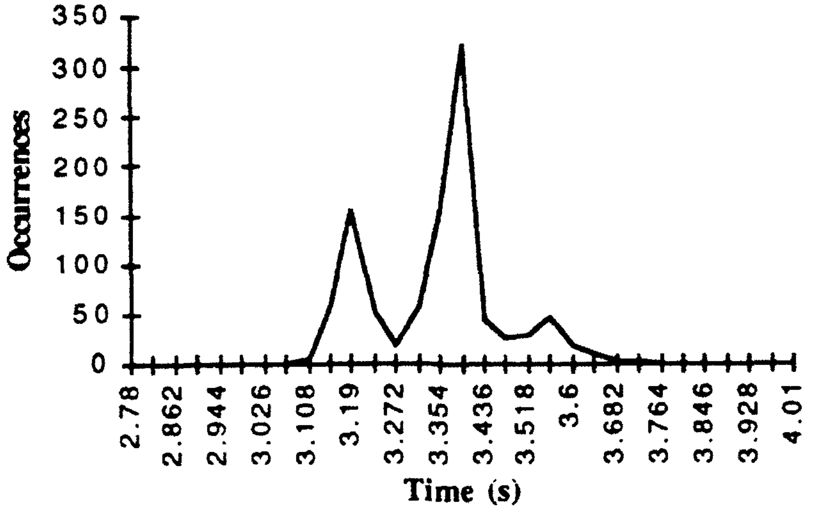


Figure G.29 Test #3, Group #1, medium file on FDDI

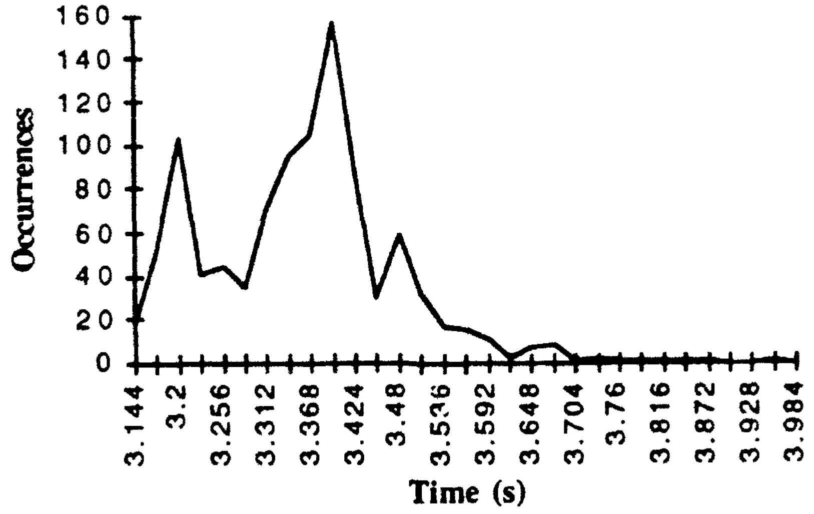


Figure G.30 Test #3, Group #2, medium file on FDDI

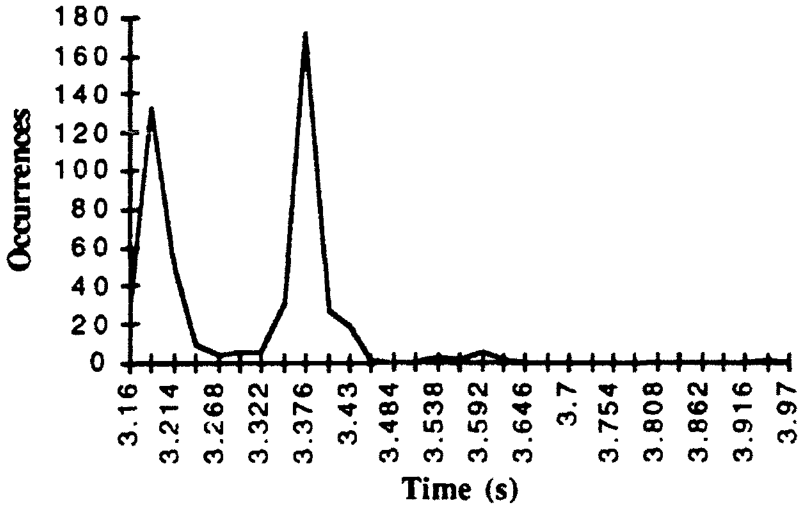


Figure G.31 Test #3, Group #3, medium file on FDDI

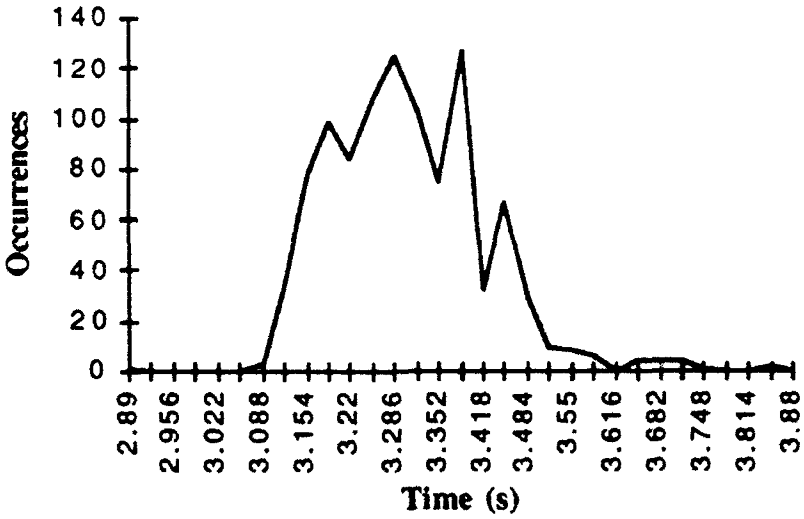


Figure G.32 Test #3, Group #4, medium file on FDDI

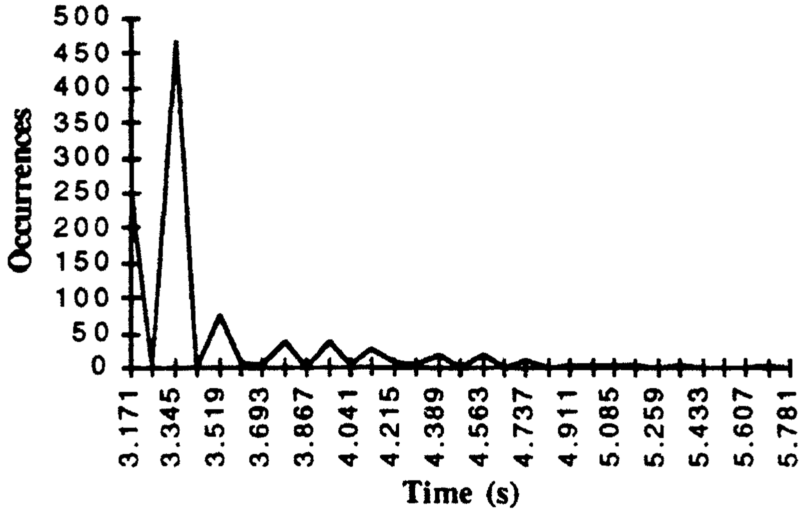


Figure G.33 Test #3, Group #1, small file on FDDI

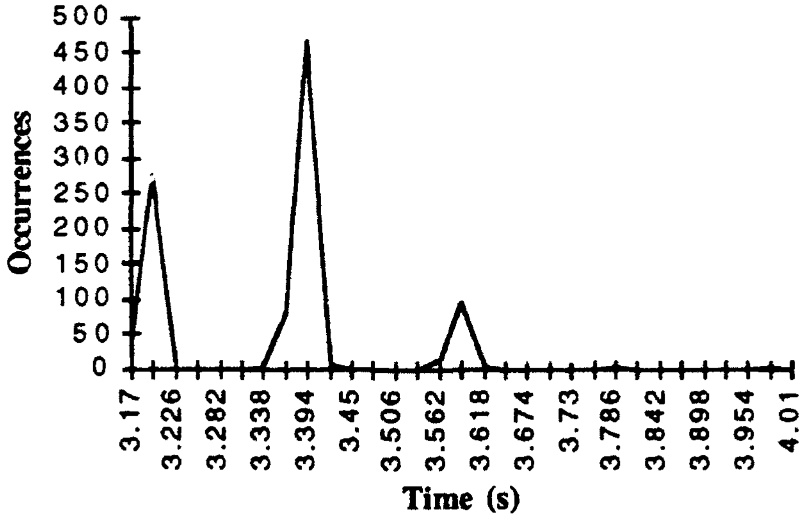


Figure G.34 Test #3, Group #2, small file on FDDI

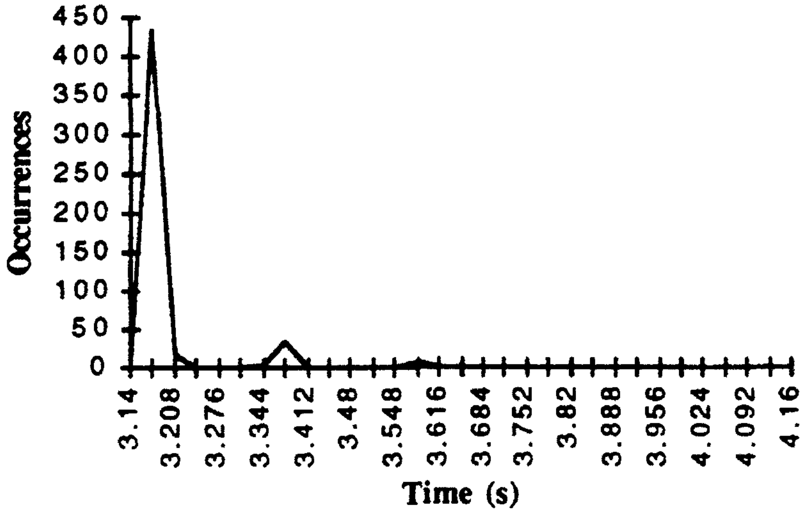


Figure G.35 Test #3, Group #3, small file on FDDI

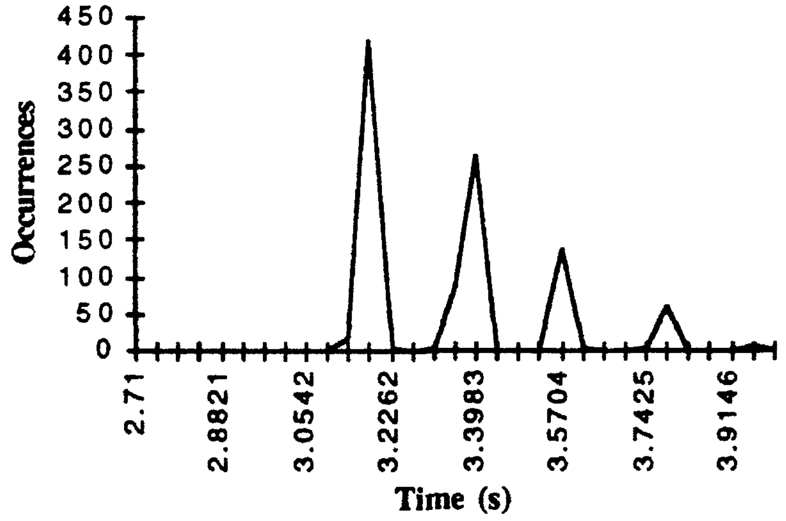


Figure G.36 Test #3, Group #4, small file on FDDI

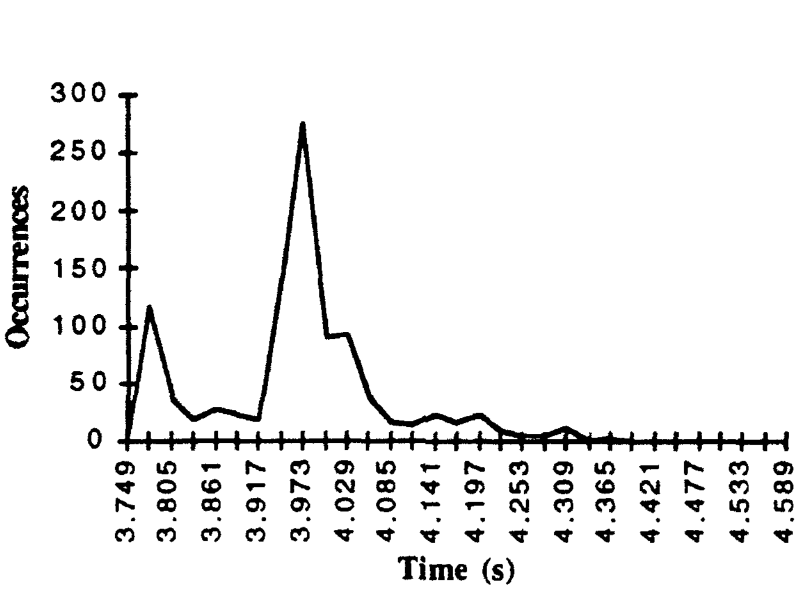


**Figure G.37 Test #4, Group #1, large file on FDDI**

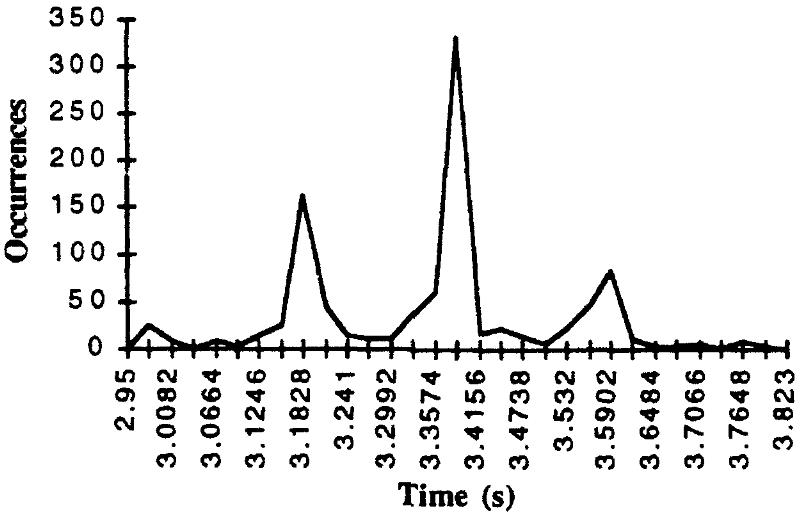


**Figure G.38 Test #4, Group #2, large file on FDDI**

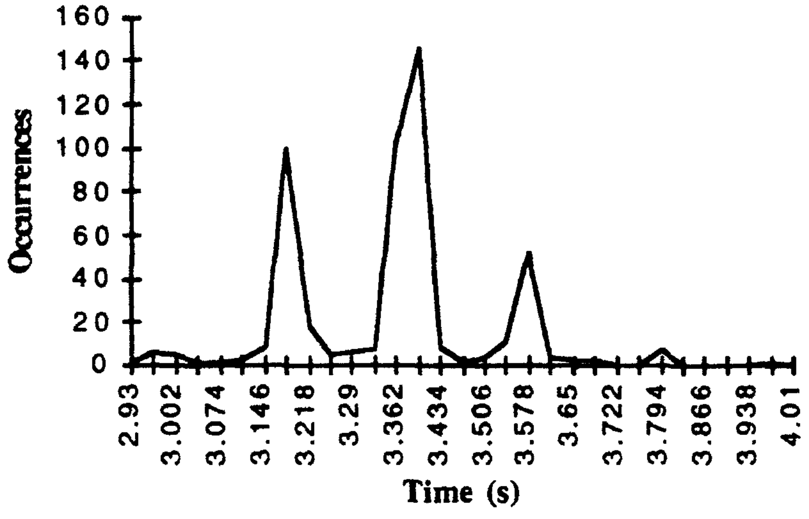


Figure G.39 Test #4, Group #3, large file on FDDI

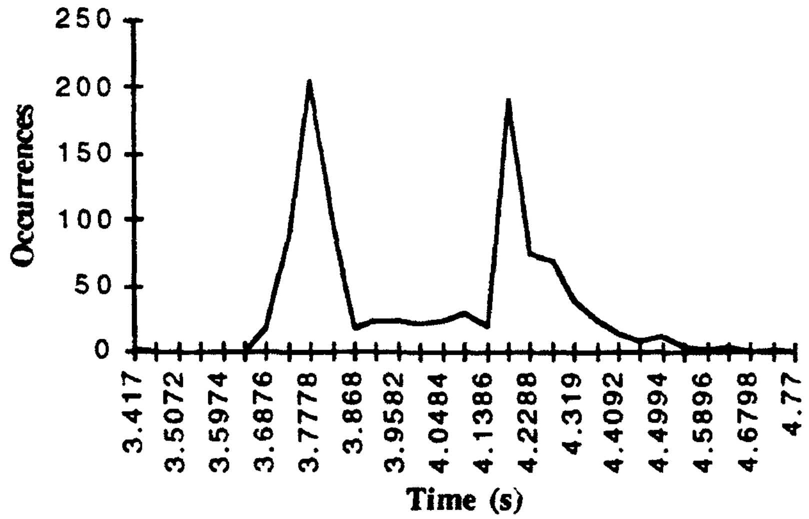


Figure G.40 Test #4, Group #4, large file on FDDI

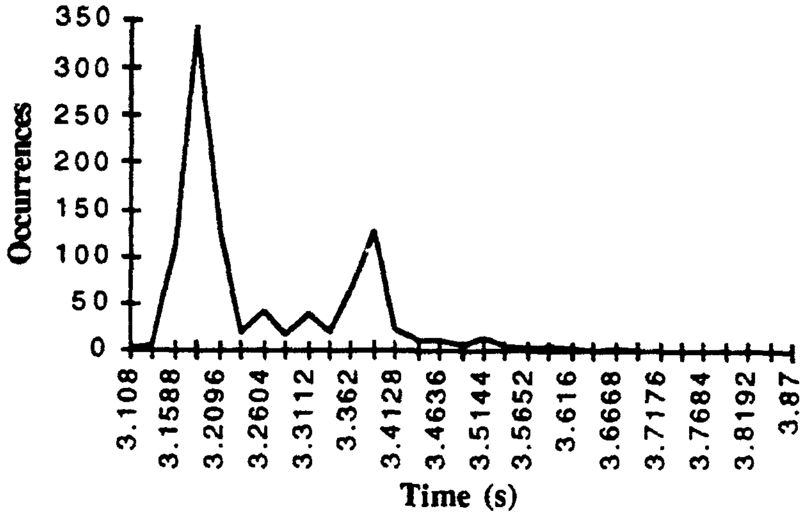


Figure G.41 Test #4, Group #1, medium file on FDDI

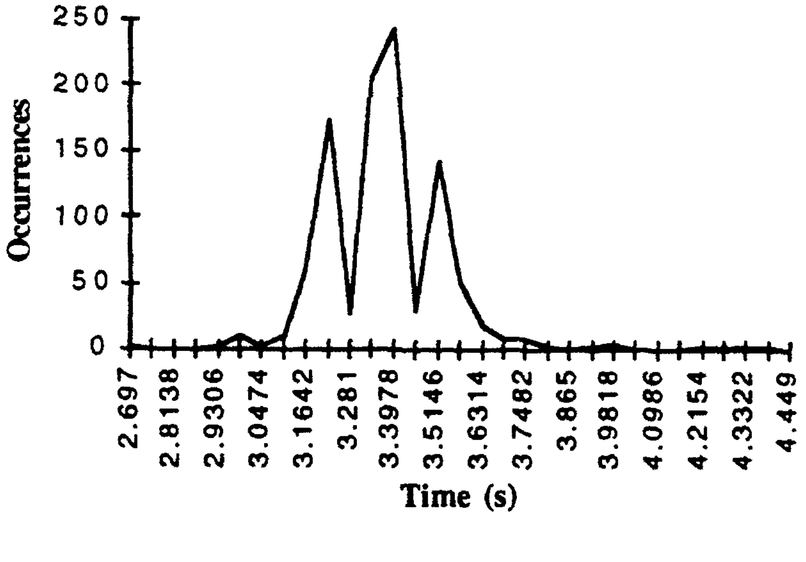


Figure G.42 Test #4, Group #2, medium file on FDDI

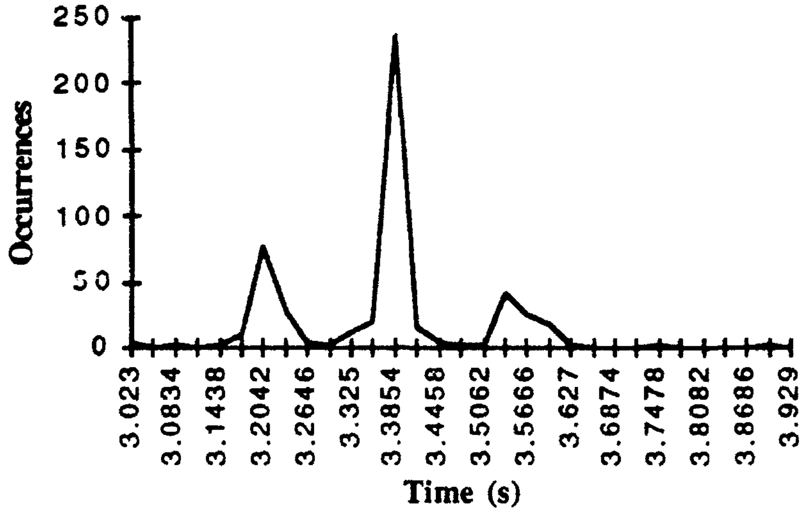


Figure G.43 Test #3, Group #3, medium file on FDDI

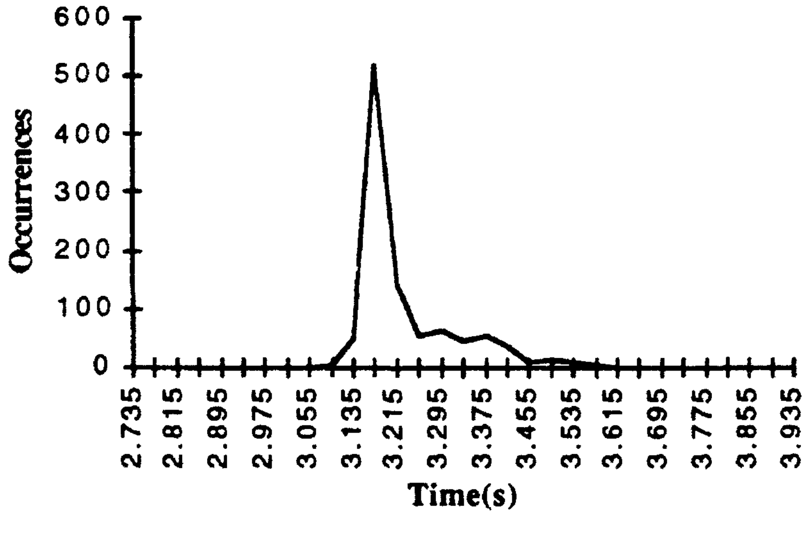


Figure G.44 Test #4, Group #4, medium file on FDDI

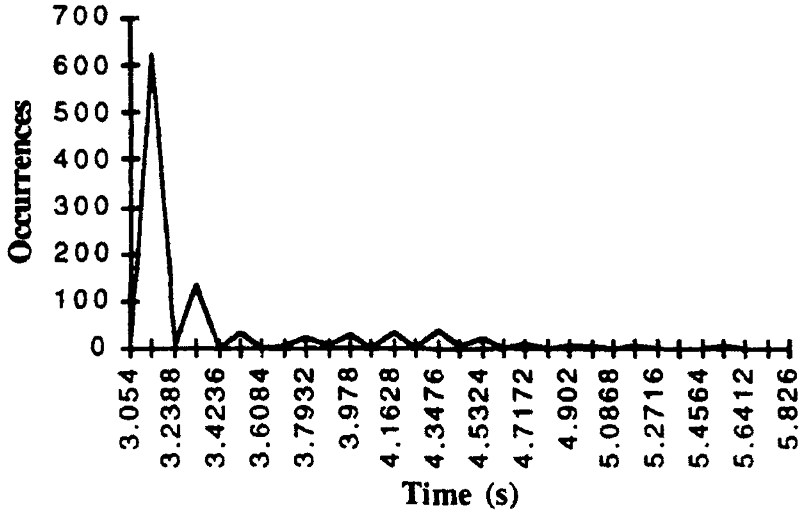


Figure G.45 Test #4, Group #1, small file on FDDI

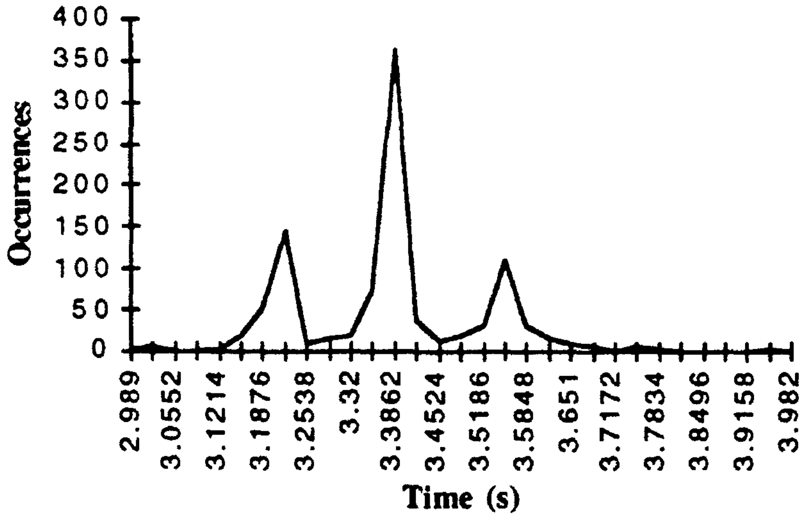


Figure G.46 Test #4, Group #2, small file on FDDI

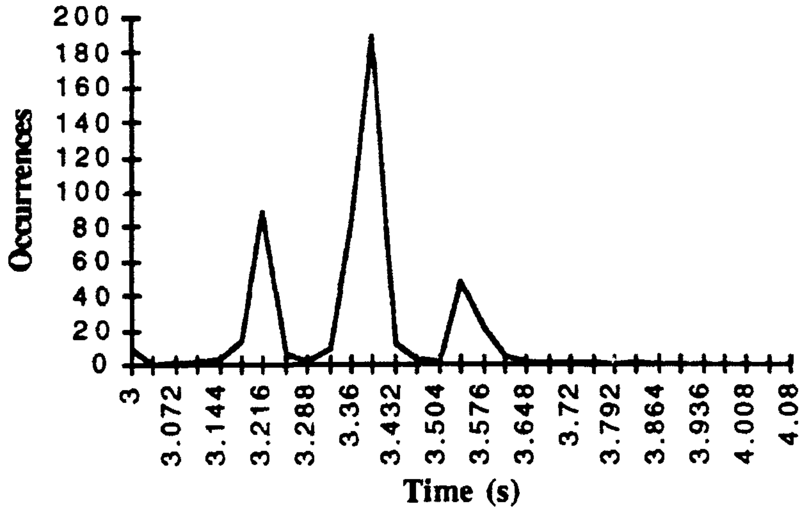


**Figure G.47 Test #4, Group #3, small file on FDDI**

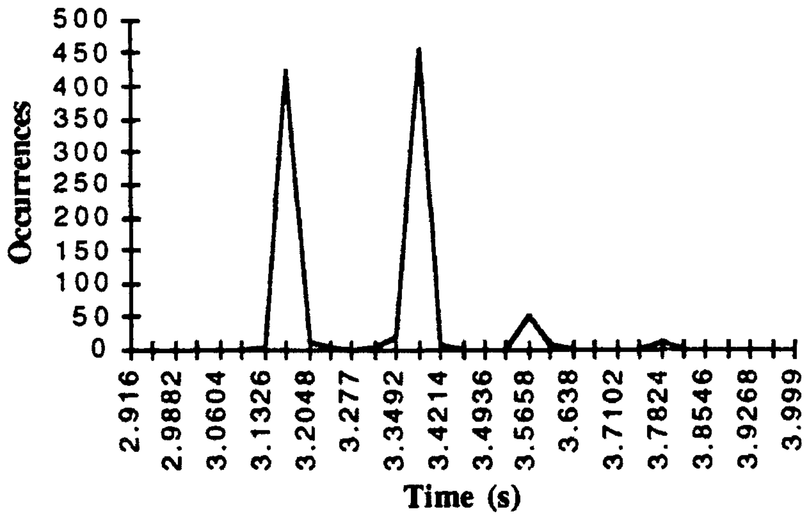


**Figure G.48 Test #4, Group #4, small file on FDDI**

# LIST OF REFERENCES

[1] Clements, Alan, *Microprocessor System Design*, PWS–Kent, Boston, MA,1987.

[2] Tanenbaum, Andrew S., *Computer Networks*, Prentice–Hall, Englewood Cliffs, NJ,1989.

[3] Comer, Douglas E., *Internetworking With TCP/IP Vol I: Principles, Protocols, and Architecture*, Prentice–Hall, Englewood Cliffs, NJ,1991.

[4] Stallings, William, *Data and Computer Communications*, Macmillan Publishing Company, New York, NY,1988.

[5] *Draft FDDI–Single Mode Fiber, Physical Layer Medium Dependent (SMF–PMD)*, X3.184–199x, American National Standards Institute, New York, NY, 30 July 1990.

[6] *Fiber Distributed Data Interface (FDDI)–Token Ring Physical Layer Protocol (PHY)*, ANSI X3.148.1988, American National Standards Institute, New York, NY, 30 June 1988.

[7] *Working Draft Proposed American National Standard FDDI Physical Layer Protocol (PHY–2)*, X3T9/91–X3T9.5/88–148 Rev 4.1, American National Standards Institute, New York, NY, 5 March 1991.

[8] *Fiber Distributed Data Interface (FDDI)–Token Ring Media Access Control (MAC)*, ANSI X3.139.1987, American National Standards Institute, New York, NY, 5 November 1986.

[9] *Working Draft Proposed American National Standard FDDI Media Access Control (MAC–2)*, X3T9/90–X3T9.5/88–139 Rev 4.0, American National Standards Institute, New York, NY, 20 October 1990.

[10] *Draft Proposed American National Standard FDDI Hybrid Ring Control (HRC)*, X3.186–199x, American National Standards Institute, New York, NY, 15 October 1991.

[11] *Preliminary Draft Fiber Distributed Data Interface (FDDI)–Station Management (SMT)*, X3T9/90–X3T9.5/84–49 Rev 6.2, American National Standards Institute, New York, NY, 18 May 1990.

[12] *SunOS Reference Manual*, Sun Microsystems, Sunnyvale, CA, 21 January 1990.

[13] *System Services Overview*, Sun Microsystems, Sunnyvale, CA, 27 March 1990.

[14] Sun FDDI/DX Software and Hardware Installation Guide.

[15] LANplex 5004 and 5012 Installation Guide.

[16] M68000 8-/16-/32-Bit Microprocessors User's Manual, Sixth Edition, Prentice Hall, Englewood Cliffs, NJ, 1989.

[17] Annamalai, K., "FDDI Physical Layer Implementation Considerations," *Proceedings of SPIE Fiber Optic Datacom and Computer Networks*, vol. 991, International Society for Optical Engineering, Bellingham, WA, 1988.

[18] Kimball, Robert M., "Optical Performance Models for FDDI Links," *Proceedings of SPIE Fiber Networking and Telecommunications*, vol. 1179, International Society for Optical Engineering, Bellingham, WA, 1989.

[19] Kahn, Ashfaq R., "FDDI Interoperability Between Vendors," *Proceedings of SPIE Fiber Networking and Telecommunications*, vol. 1179, International Society for Optical Engineering, Bellingham, WA, 1989.

[20] Paige, Jeffrey L. and Howard, Edward A., "SAFENET II–The Navy's FDDI-based Computer Network Standard," *Proceedings of SPIE Campus-Wide, and Metropolitan Area Networks*, vol. 1364, International Society for Optical Engineering, Bellingham, WA, 1990.

[21] Coden, Michael H., Bulusu, Dutt V., Ramsey, Brian, Szutka, Edward and Morrow Joel, "Modular FDDI Bridge and Concentrator," *Proceedings of SPIE Campus-Wide, and Metropolitan Area Networks*, vol. 1364, International Society for Optical Engineering, Bellingham, WA, 1990.

[22] Stevens, R. Scott, "FDDI Network Cabling," *Proceedings of SPIE Campus-Wide, and Metropolitan Area Networks*, vol. 1364, International Society for Optical Engineering, Bellingham, WA, 1990.

[23] AlKhatib, Hasan S.. "A Vertically Integrated Transport and Media Access Protocol for Gbps Local Area Networking," *Digest of Papers, Compcon Spring 92*, IEEE Computer Society Press, Los Alamitos, CA 1992.

[24] Sun Press Release, "Fiber Distributed Data Interface/SBus (FDDI/S)," SMCC Worldwide Product Announcement Information, 7 October 1991.

[25] Personal communication with H. AlKhatib, 4 May 1992 17:47 PST.

[26] Lundy, G. M., "Improving Throughput in the FDDI Token Ring Network," *Protocols for High Speed Networks, II, North*-Holland, New York, NY, 1991.

[27] Ahamad, Mustaque, Ammar, Mostafa H., Bernabéu-Aubán, José M., Khalidi, M. Yousef, "Using Multicast Communication to Locate Resources in a LAN-Based Distributed Systems," *Proceedings of IEEE 13th Conference on Local Computer Networks*, Washington, DC, October 1988.

[28] Belkeir, Nasr E., Ahamad, Mustaque, "Low Cost Algorithms for Message Delivery in Dynamic Multicast Groups," *Proceedings of IEEE Ninth Conference on Distributed Computing*, June 1989.

[29] Hughes, Larry, "Identifying Migrated Objects Using Multicast Addresses," *Computer Communications*, September 1991.

[30] Green, Larry, "Performance Analysis of FDDI," *Digest of Papers, Compcon Spring 87*, IEEE Computer Society Press, Washington, DC 1992.

[31] McIntosh, Thomas F., "Engineering Building and Campus networks for Fiber Distributed Data Interface (FDDI)," *Proceedings of SPIE Fiber Networking and Telecommunications*, vol. 1179, International Society for Optical Engineering, Bellingham, WA, 1989.

[32] Cohn, Marc, "A Lightweight Transfer Protocol for the U.S. Navy SAFENET Local Area Network Standard," *Proceedings of IEEE 13th Conference on Local Computer Networks*, Washington, DC, October 1988.

[33] Weaver, A. C., and Colvin, M. A., "A Real-Time Messaging System for Token Ring Networks," *SOFTWARE - Practice and Experience*, Vol. 17(12), December 1987.

[34] Colvin, M. A., and Weaver, A. C., "Performance of Single Access Classes on the IEEE Token Bus," *IEEE Transactions on Communications*, Vol. COM-34, Number 12, December 1986.

[35] Strayer, W. T., and Weaver, A. C., "Performance Measurement of Data Transfer Services in MAP," *IEEE Networks*, May 1988.

[36] Strayer, W. T., and Weaver, A. C., "Is XTP Suitable for Real-Time Distributed Systems?", *International Workshop on Advanced Communications and Applications for High Speed Networks*, Munich, Germany, March 16-19, 1992.

[37] Sankar, R., and Yang, Y. Y., "Performance Analysis of FDDI," *Proceedings of IEEE 14th Conference on Local Computer Networks*, Los Alamitos, CA, October 1989.

[38] Tarrant, Peter and Truman, Alan, "Implementation of FDDI in the Intelligent Wiring Hub," *Proceedings of SPIE Campus-Wide, and Metropolitan Area Networks*, vol. 1364, International Society for Optical Engineering, Bellingham, WA, 1990.

[39] AlKhatib, Hasan S., "High Performance Local Area Networks," COMPCON Spring 1992 Conference, Los Alamitos, CA, February, 1992.

# INITIAL DISTRIBUTION LIST

| | |
|---|---|
| Dudley Knox Library<br>Code 52<br>Naval Postgraduate School<br>Monterey, CA 93943 | 2 |
| Chairman, Code CS<br>Computer Science Department<br>Naval Postgraduate School<br>Monterey, CA 93943 | 2 |
| Chairman, Code EC<br>Department of Electrical and Computer Engineering<br>Naval Postgraduate School<br>Monterey, CA 93943 | 2 |
| Professor Luqi, Code CS/Lq<br>Computer Science Department<br>Naval Postgraduate School<br>Monterey, CA 93943 | 10 |
| Professor Shridhar Shukla, Code EC/Sh<br>Department of Electrical and Computer Engineering<br>Naval Postgraduate School<br>Monterey, CA 93943 | 1 |
| Defense Technical Information Center<br>Cameron Station<br>Alexandria, VA 22304–6145 | 2 |
| Commandant (G–TPR)<br>U. S. Coast Guard<br>Washington, DC 20593 | 2 |
| Commandant (G–PIM–3)<br>U. S. Coast Guard<br>Washington, DC 20593 | 1 |
| Sun Microsystems, Inc.<br>1842 N. Shoreline Blvd.<br>Mountain View, CA 94043<br>Attn: Jean Chappelle | 1 |

| | |
|---|---|
| Synemetics Inc.<br>4017 Clipper Court<br>Bayside Business Park<br>Fremont, CA 94538<br>Attn: George Ducharme | 1 |
| Ada Joint Program Office<br>OUSDRE (R&AT)<br>The Pentagon<br>Washington, DC 20301 | 1 |
| Office of Naval Technology<br>Code 227<br>Attn. Dr. Elizabeth Wald<br>810 N. Quincy<br>Arlington, VA 22132-5000 | 1 |
| Office of Naval Research<br>Computer Science Division, Code 1133<br>Attn. Dr. Van Tilborg<br>800 N. Quincy<br>Arlington, VA 22217-5000 | 1 |
| Office of Naval Research<br>Computer Science Division, Code 1133<br>Attn. Dr. R. Wachter<br>800 N. Quincy<br>Arlington, VA 22217-5000 | 1 |
| University of California at Berkeley<br>Department of Electrical Engineering and Computer Science<br>Computer Science Division<br>Attn. Dr. C. V. Ramamoorthy<br>Berkeley, CA 90024 | 1 |
| Attn. Dr. R. Wachter<br>United States Laboratory Command<br>Army Research Office<br>P. O. Box 12211<br>Research Triangle Park, NC 27709-2211 | 1 |

|  |  |
|---|---|
| G. A. Hammar<br>501 Privateer Rd.<br>North Palm Beach, FL 33408 | 10 |
| Naval Research Laboratory<br>Code 8140<br>Attn. Dr. Lou Chmura<br>4555 Overlook Ave.<br>Washington, DC 20375-5000 | 1 |