Data Communication Network at the ASRM Facility

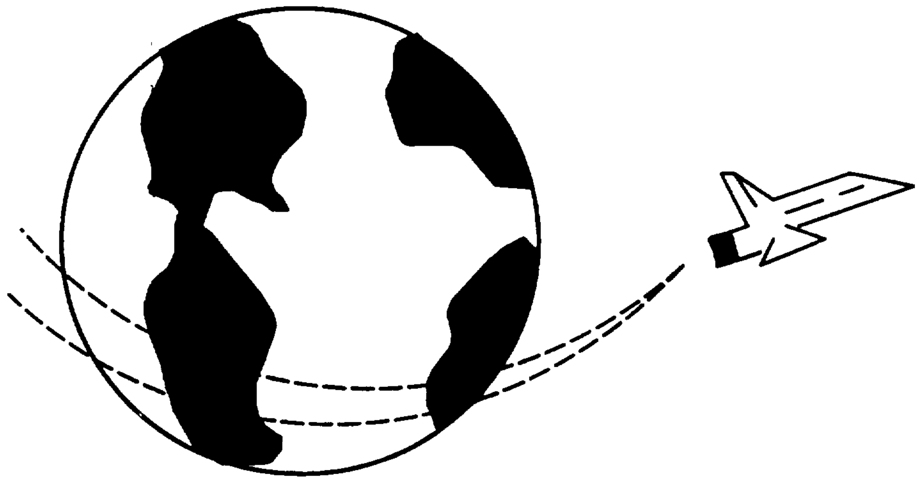
Second Year Final Report

February 5, 1993

[illegible]
P. 89

Robert J. Moorhead II
Wayne D. Smith
Principal Investigators

Ravi Nirgudkar
Zhifan Zhu
Research Assistants



For the period : February 6, 1992 to February 5, 1993

Department of Electrical and Computer Engineering
Mississippi State University
Mississippi State, MS 39762
(601) 325 – 2850

NASA GRANT NAG8 – 866

Submitted to : **W. W. Robinson / EB32**

NASA / MSFC

ACKNOWLEDGEMENTS

Most of the information on the implementation details herein was obtained from Merlin Hill (AAD), Dale Scruggs (RUST), Kirk Weiss (RUST), and John Donaldsen (LMSC). Their assistance in providing information is graciously acknowledged.

The assistance of our NASA Technical Officer, Walter Robinson, is likewise acknowledged. Without his help, this study would be going nowhere.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

The main objective of the report is to present the overall communication network structure for the Advanced Solid Rocket Motor (ASRM) facility being built at Yellow Creek near Iuka, Mississippi *as of today*. This report is compiled using information received from NASA/MSFC, LMSC, AAD, and RUST Inc. [1–8, 28].

As per the information gathered, the overall network structure will have one logical FDDI ring acting as a backbone for the whole complex. The buildings will be grouped into two categories viz. manufacturing critical and manufacturing non–critical. The manufacturing critical buildings will be connected via FDDI to the Operational Information System (OIS) in the main computing center in B_1000. The manufacturing non–critical buildings will be connected by 10BASE–FL to the Business Information System (BIS) in the main computing center. The workcells will be connected to the Area Supervisory Computers (ASCs) through the nearest manufacturing critical hub and one of the OIS hubs.

The network structure described in this report will be the basis for simulations, to be carried out next year. The Comdisco's Block Oriented Network Simulator (BONeS) [10] will be used for the network simulation.

The main aim of the simulations will be to evaluate the loading of the OIS, the BIS, the ASCs, and the network links by the traffic generated by the workstations and workcells throughout the site.

1. INTRODUCTION

The Advanced Solid Rocket Motor (ASRM) project at Yellow Creek near Iuka, Mississippi is part of a National Aeronautics and Space Administration (NASA) program to substantially improve the flight safety, reliability, productivity, and performance of the space shuttle's solid rocket motors. The ASRM is a replacement for the current space shuttle Redesigned Solid Rocket Motor (RSRM).

The facility will be government–owned but contractor–operated. Lockheed Missiles and Space Company Inc., ASRM division (LMSC) is the prime contractor. The operation of the facility will be directed by the subcontractor Aerojet ASRM division (AAD); RUST International Corporation (RUST) is responsible for the engineering and construction of the facility.

The development of the ASRM is expected to take about six years, with the first new motors planned for a shuttle flight in 1996.

1.1 ASRM Communication Network Structure :

The operations at the ASRM site will be performed in different buildings scattered over a large area. These buildings will be connected to each other through a Local Area Network (LAN).

The buildings are classified as Manufacturing Critical buildings and Manufacturing Non–Critical buildings, depending on the type of operation performed within the building. There will be five Manufacturing Critical hubs and five Manufacturing Non–Critical hubs connecting the respective buildings to the Main Computing Center in Building 1000 (B_1000). All the workcells will be connected to the nearest Manufacturing Critical Hub.

Each Manufacturing Critical hub will be communicating with either the Operational Information System (OIS) or an Area Supervisory Computer (ASC) by Fiber Distributed Data Interface (FDDI) protocol over an optical fiber link. The workstations will be interacting with the OIS, while the workcell's data will be routed to ASCs.

Each Manufacturing Non–Critical hub will be communicating with the Business Information System (BIS) by Ethernet protocol over an optical fiber link.

The two VAXEN in the OIS VAX cluster can communicate directly with each other and can even perform load sharing if required. The BIS on the other hand is a single entity.

All the printer jobs throughout the campus will be routed through the Gandalf Terminal Server by the BIS.

For all this data transfer, the required routing, security, and flexibility will be provided by the Cabletron Multi Media Access Centers (MMACs) which will be used throughout the campus.

The overall network logically forms one big FDDI ring although physically it looks like a combination of various point–to–point connections.

1.2 Summary Of The Forthcoming Chapters :

The main aim of this report is to present the overall communication network structure for the ASRM facility. The report is composed of chapters discussing the Main Computing Center (B_1000), the Network Cabling, and the Protocols to be used on the various links

The chapter on 'Main Computing Center' concentrates on the network connection inside B_1000. The chapter explains the OIS, the BIS, the ASCs, and the Cabletron MMACs interconnection.

The chapter on 'Network Cabling' summarizes the Outdoor Cabling and the Indoor Cabling at the ASRM site. The chapter also describes the Telephone Cabling in brief.

The chapter on 'Protocols' gives an overview of the different protocols that will be used on different links in the network. The chapter also gives a brief theoretical background for each of the protocol and explains the 'FDDI Dual Ring of Trees'.

The 'Conclusion' chapter at the end of the report comments on the network expectations, the network evaluation parameters, the network modeling, the data rates for the workstations, and the workcells. Lastly some preliminary conclusions are drawn at the end of that chapter.

The mid–year report submitted to NASA/MSFC is included in this report as Appendices A5, A6, and A7.

Appendix A5 is an investigation of X window traffic using TCP/IP over Ethernet networks. Appendix A6 is a survey study of performance concepts of X window applications with Macintosh computers. Appendix A7 is a tutorial on DECnet protocols.

1.3 Research Objective :

Now that the design of the long–term network is being finalized, the main objective of the research will be to simulate and analyze the network to determine its performance under different load conditions.

The Comdisco's Block Oriented Network Simulator (BONeS) will be used for network simulation. The performance of the network with the given topology and protocols can be evaluated, using BONeS. The two primary evaluation parameters that will be used to judge the network performance will be the throughput and the delay.

The aim of the simulations will be to look into the loading of the OIS, the BIS, the ASCs, and the network links due to the traffic generated by the workstations and the workcells over the entire site.

2. MAIN COMPUTING CENTER (BUILDING 1000)

2.1 Purpose Of The Main Computing Center :

Building 1000 (B_1000) is to provide an efficient means to plan and control the manufacturing of solid rocket motors for the ASRM project. All the workstations and the workcells communicate only with the OIS, the BIS, and the ASCs in B_1000; there is no peer–to–peer communication required. B_1000 also provides a link between the business functions and the manufacturing functions of the facility. The interconnection between the devices in B_1000 is shown in Figure 2.1.

The OIS will be a VAX cluster consisting of two VAX 6000 each with one FDDI adapter.

Each of the ASCs will be a VAX 4400 with an Ethernet adapter. Currently LMSC is planning on two ASCs, but hoping to be able to get by with only one.

B_ 1000 will also have a Gandalf Terminal Server. The Gandalf Terminal Server is a giant terminal server with a multitude of ports. The Gandalf can support 12 separate Ethernet channels. The Gandalf Terminal Server physically is a cabinet 6 feet tall. It will be the *only* terminal server throughout the campus.

The OIS, the BIS, the ASCs, and the Gandalf terminal server in B_1000 will be connected to the outside network complex by the Cabletron MMACs, the intelligent hubs. There will be fiber connection between the OIS, and the Cabletron hubs. There will be a copper connection between the BIS and its Cabletron hub, and between the ASC, and their Cabletron hub. The Cabletron hubs provide the necessary security, routing, and redundancy.

In addition to the devices mentioned, Building 1000 will have nine more Cabletron hubs distributed in three switch rooms viz. 1243, 1245, and 1250. B_1000 will also have approximately 100 printers, 36 CAD workstations, 400 MacIntosh computers connected to the BIS hub by 10BASET, 50 PCs connected to the BIS hub by 10BASET, 40 Engineering workstations on 10BASE2, 289 dumb terminals connected via ADI to the Gandalf to the BIS hub, and a connection to the PSCNI router. The BIS hub connections are shown in Appendix A4. The exact number, the configuration, and the

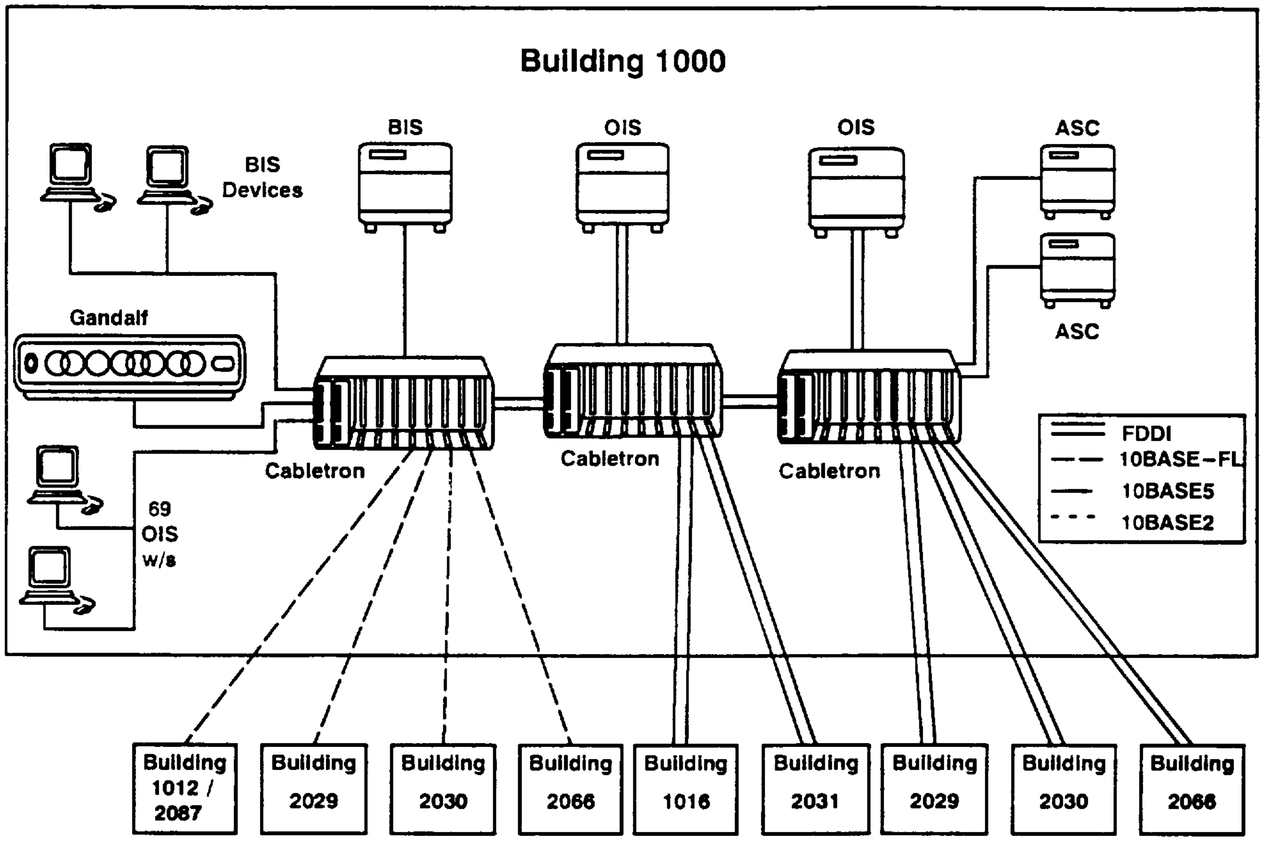


Figure 2.1 Inside Building 1000

communication requirements of each must be determined before significant simulations can commence. It appears that within the last three months the function of the BIS has changed significantly.

2.2 Cabletron Devices :

All the Cabletron devices used will be Multi Media Access Centers (MMACs) with different add–on Media Interface Modules (MIM). For detailed information refer to Appendix A2.

The main functions of the Cabletron devices will be as follows :

1. Routing of the printer jobs to the Gandalf Terminal Server through the BIS.

2. Routing of the accumulated data from the workcells to the ASCs.

3. Providing the load sharing capability for the two OIS VAX.

4. Providing connectivity to the BIS or the OIS for various devices.

All the MMAC devices used will be the MMAC–8FNBs, and the first slot in each of the MMAC devices will be a EMME multichannel management / bridge module.

2.3 BIS Network :

The BIS is a VAX 3400 VAX cluster consisting of three VAX 6000s (one 6410 and two 6420s) and five different console systems.

The BIS will be connected to its Cabletron hub by thick–wire coaxial with 10BASE5 protocol.

The main functions of the BIS network will be as follows

1. Routing to / from the Gandalf Terminal Server.

2. Serving the most of the devices inside Building 1000.

3. Serving all the Manufacturing Non–Critical workstations.

The BIS hub and one of the OIS hubs will be connected to provide a path for the printer jobs from the Manufacturing Critical buildings to the Gandalf Terminal Server and subsequently to the printers. Except for this function and the 69 unmasked workstations, all the BIS traffic will be independent of the OIS traffic.

2.4 OIS Network :

The OIS in B_1000 is a VAX cluster consisting of two VAX 6000 each with FDDI adapters.

Each OIS VAX will be connected to a Cabletron hub by optical fiber using the FDDI protocol.

The main operations of the OIS will be as follows :

1. To provide efficient means to plan, control, and provide data collection using commercial software packages [2].

2. To download and upload information to each ASC, which will serve a group of workcells. The ASCs will help isolate the traffic from the OIS.

2.5 ASC Network :

The two ASCs will be two VAX 4400, each with an Ethernet adapters.

The ASCs will be connected to the Cabletron hub by thick wire coaxial cable using the 10BASE5 protocol.

The ASC is a real–time device which handles the Application Program Interfaces (APIs). The ASC has similar functionality to the OIS. The ASC will control and manage a set of workcells. The OIS and ASCs will be connected through a self–configuring bridge. The bridge separates the traffic between the shop floor and the OIS. The ASCs will communicate with the OIS occasionally with a large block of data, rather than communicating continuously, which would slow down the OIS operation. Also if the OIS goes down, the ASC will keep the network alive and log the data.

3. NETWORK CABLING

3.1 Transmission Media :

The transmission medium is the physical path between transmitter and receiver in a data transmission system. The characteristics and quality of data transmission are determined both by the nature of the signal and the nature of the medium. Table 3.1 gives typical characteristics for guided media, including the total data rate that the medium can support, the bandwidth the medium can transmit, and the required repeater spacing for digital transmission [12].

Transmission Medium	Total Data Rate	Bandwidth	Repeater Spacing
Twisted Pair	4 Mbps	250 KHz	2 – 10 Km
Coaxial Cable	500 Mbps	350 MHz	1 – 10 Km
Optical Fiber	2000 Mbps	2000 MHz	10 – 100 Km

Table 3.1 Transmission Media Characteristics

3.1.1 Twisted Pair :

Twisted pair cable is made up of copper wire twisted together in precisely calculated fashion, since the wires carry more information further when twisted together than when left as separate strands. For Local Area Networks (LANs) two varieties of twisted pair cables are used, *shielded* and *unshielded.*

3.1.2 Coaxial Cable :

Coaxial cable contains a single central wire, which is surrounded by special insulation and wire mesh. There are many types of coaxial cables; coaxial carries more information further and faster than twisted pair cable.

3.1.3 Optical Fiber :

Optical fiber cable contains a hair thin strand of optical fiber surrounded by special shielding and insulation. Optical fibers carry pulses of light, instead of bursts of electricity. With the right supporting hardware, optical fiber can carry more information further in less space than any other type of cable.

3.1.4 Cabling at ASRM site :

At the ASRM site all the outdoor cabling and much of the indoor cabling will be optical fiber. Thin–wire coaxial and thick–wire coaxial will be used in B_1000. Twisted pair will be used in manufacturing non–critical buildings.

3.2 Outdoor Cabling At The ASRM Site :

All the outdoor cabling will be optical fiber. All systems will be supported with 62.5 / 125 micron multimode optical fiber. The outdoor cabling will support the FDDI standards for installation methodology and signal loss. There will be no outside splicing of the fiber, while all the indoor splicing will be done by fusion.

Every FDDI hub will have at least three redundant paths, viz. Channel A and Channel B of FDDI and a 10BASE–FL backup. Also every FDDI hub will have two redundant dual rings.

3.2.1 Manufacturing Critical Buildings Connections :

The manufacturing critical buildings will have two FDDI data paths from the B_1000 with automatic switchover. One data path will be buried, while the other will be aerial. Buildings **1016, 2029, 2030, 2031,** and **2066** are the manufacturing critical buildings, each will have a hub directly connected to a hub in B_1000. Buildings 2060 and 2076 will be connected to the hub in building 2029. Each hub will receive two pair of fibers from the outside cable plant. All workstations and workcell devices will receive two fibers each from the respective hub.

3.2.2 Manufacturing Non–Critical Buildings Connections :

All the manufacturing non–critical buildings in the complex will receive two fibers for its hub from the outside cable plant. All the workstations inside the buildings will get two fibers each from the respective hub.

3.3 Indoor Cabling At The ASRM Site :

For the indoor cabling in the manufacturing critical buildings, the 10BASE–FL protocol will be used, mainly because it allows lower light levels and 16 redundant data paths, and because the Cabletron devices support the 10BASE–FL protocol.

3.4 Telephone Cabling :

The telephone system at ASRM Iuka is being built around an Intecom S/80 switch. The only overlap in the voice and data networks now is between the Gandalf terminal server and the RS–232 devices it serves. The Intecom system will be used as the network for communicating serial information between the aforementioned devices. The RS–232 devices consist of printers and some workcells. The Intecom network's characteristics were studied and reported last year [19].

3.5 Physical Distances :

Table 3.2 gives the physical distance of all the buildings from the nearest hub. Table 3.3 gives the physical distance of all the hubs from B_1000.

Building No.	Building Name	No. of workstns	No. of workcells	Link	Nearest Hub	Distance from hub(feet)
1000	Engineering / Computer	(69+ 40+36)	00	—	1000	—
1001	Security and Medical	03	00	Link #4	1000	1500
1010	Central Warehouse	10	00	Link #4	1000	600
1012	Warehouse 'A'	14	00	Link #7	1012	—
1016	Case Prep. and Refurbishment	27	16	Link #3	1016	—
1022	Chemical Storage	01	00	Link #4	1000	2600
1025	Carpenters Shop	01	00	Link #4	1000	2400
1032	Office	03	00	Link #5	2066	800
1045	Training Center	04	00	Link #4	1000	1400
2015	Pre – Mix (Mix / Cast)	00	00	Link #1	2029	1450
2028	Tool Clean / Core Prep.	08	00	Link #1	2029	2600
2029	Remote Control Room	12	01	Link #1	2029	—
2030	Non Destructive Evaluation Facility	03	02	Link #6	2030	—
2031	Final Assembly	15	00	Link #2	2031	—
2042	Main Motor Storage	04	00	Link #6	2030	8650
2060	Small Scale Propellant Proc.	04	02	Link #1	2029	2250
2066	Quality Assurance Lab.	05	01	Link #5	2066	—
2070	Sample Preparation	01	00	Link #5	2066	1450

Building No.	Building Name	No. of workstns	No. of workcells	Link	Nearest Hub	Distance from hub(feet)
2076	Qualification Motor Facility	04	01	Link #1	2029	2550
2082	HTPB Storage Tank Farm	01	00	Link #1	2029	850
3003	Deload – Open area No Building	00	01	Link #6	2030	7050
3005	Control Building	03	01	Link #6	2030	5950
3010	Incinerator System Building	00	01	Link #6	2030	4950
3011	Feed Prep. Facility	01	01	Link #6	2030	6600
4001	Shipping Dock	01	00	Link #6	2030	9700

	TOTAL	**270**	**27**			

Table 3.2 Distances of each building from the nearest hub

Link	**Distance (feet)**	**Number of Workstations on the link.**	**Number of Workcells on the link.**	**Type of the Link**
Link # 1 : (2029)	6700	29	04	Critical and Non–critical
Link # 2 : (2031)	4650	15	00	Critical
Link # 3 : (1016)	1450	27	16	Critical
Link # 4 : (1000)	00	19	00	Non–critical
Link # 5 : (2066)	3550	09	01	Critical and Non–critical
Link # 6 : (2030)	5000	12	06	Critical and Non–critical
Link # 7 : (1012)	950	14	00	Non–Critical

Table 3.3 Distances of each hub from B_1000

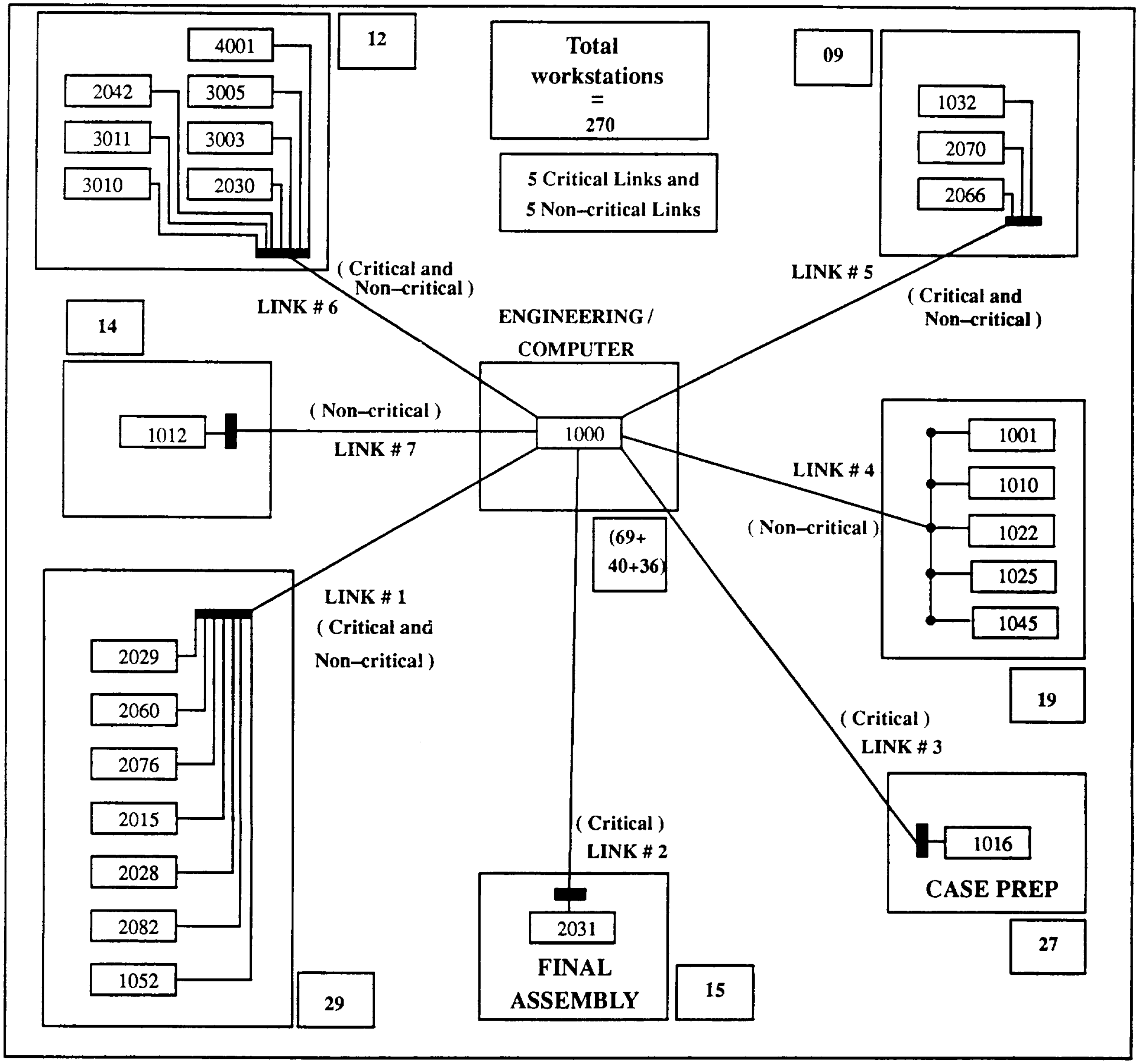


Figure 3.1 Hub Connections

4. PROTOCOLS

4.1 Standard Protocols :

For two entities to successfully communicate over a network, they must conform to some mutually acceptable set of conventions referred to as a *Protocol.* A Protocol may be defined as a set of rules governing the exchange of data between two entities. The key elements of a protocol are :

Syntax : Includes data format, coding, and signal levels.

Semantics : Includes control information for co–ordination and error handling.

Timing : Includes speed matching and sequencing.

4.1.1 IEEE 802 LAN Protocols :

The Institute of Electrical and Electronic Engineer (IEEE) has established different committees to develop standards for LANs :

802.2 Logical Link Control (LLC)

802.3 Carrier Sense Multiple Access Collision Detect (CSMA / CD)

802.4 Token Bus

802.5 Token Ring

4.1.2 ANSI LAN Protocol :

The American National Standards Institute (ANSI) has developed a specification for LANs and optical fiber. The standard is called Fiber Distributed Data Interface (FDDI) and was written by ANSI committee X3T9.5 [14].

4.1.3 Protocols At ASRM Site :

For the communication network at the ASRM site, two protocols are specified viz. FDDI and CSMA/CD.

All the manufacturing critical buildings will be connected to B_1000 by links with FDDI protocol, and all the manufacturing non–critical buildings will be connected to B_1000 by links with 10BASE–FL protocol (i.e. CSMA/CD on optical fiber). The protocol inside the manufacturing critical building will be 10BASE–FL and the protocol inside the manufacturing non–critical building will be 10BASE–T (i.e. CSMA/CD on twisted pair).

All the fiber systems will be supported with 62.5 / 125 micron multimode optical fiber. The outdoor cabling will support the FDDI standards for installation methodology and signal loss.

4.2 FDDI Protocol :

The FDDI protocol works on an optical fiber channel at 100 Mbps. Up to 1000 nodes can be placed on one optical fiber ring. The nodes can be spaced as far as 2 Km apart and the ring circumference can be up to 200 Km [14].

FDDI specifies a topology in which two independent, counterrotating optical fiber rings are in place, which provides a overall bit rate of 200 Mbps, with each channel operating at 100 Mbps.

In Figure 4.1 some devices (A type) are attached to both inner and outer rings, while other devices (B type) are attached by only one ring. This allows a user to designate those critical stations which need additional back–up and higher speeds as type A stations. The other, less important ones such as isolated workstations or low–priority terminals, can be hooked up as type B stations, at a lesser cost.

The Ring Wiring Concentrator (RWC) acts as a reconfiguration and concentration point for all optic wiring and data traffic.

The connectors into the terminals and wiring concentrator are laser diodes which can drive the fiber at a rate of over 100 MHz. FDDI stipulates a standard optic light wave of 850 nanometers.

FDDI utilizes a 4B / 5B encoded signal at a rate of 125 Mbps. Encoded signals are grouped as data and linestates. Data signals contain a nibble (4–bits) of data encoded into a 5 bit symbol, hence the resulting data rate is 4 / 5 of the actual bit rate or 100 Mbps. Linestate signals are non–data 5 bit

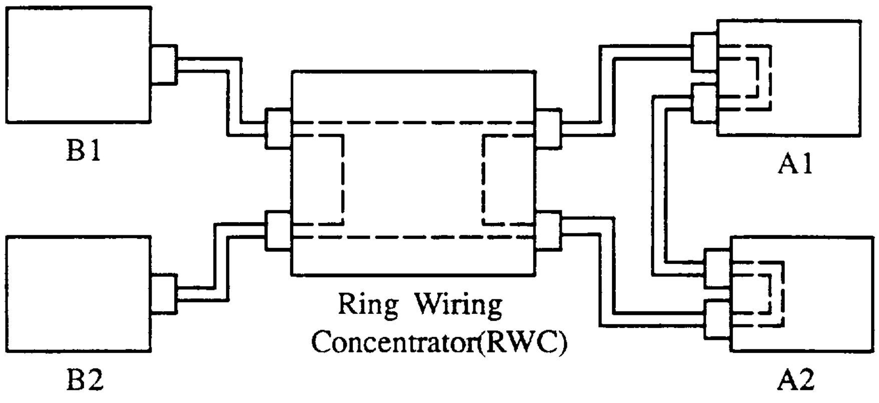


Figure 4.1 FDDI Topology

symbols that allow for a rudimentary communication protocol below the Medium Access Control (MAC) layer.

FDDI uses a multiple token passing protocol. The token circles the ring behind the last transmitted packet from a device. Any station wishing to transmit data seizes the token, removes the token, places the packet or packets on the ring, and then issues the new token directly behind the data stream.

4.3 CSMA / CD Protocol :

CSMA / CD is the most commonly used Medium Access Control (MAC) technique for bus / tree topologies, also referred to as *Listen While Talk* (LWT). The original baseband version of this technique was developed and patented by XEROX.

In CSMA / CD many different stations are connected to a common bus. If two stations try to transmit at the same time then the packets will collide, at which point each station waits a random amount of time before trying to transmit again. If the packet from a station collides again, then the station waits a longer amount of time, determined by the random exponential backoff time for that station, before trying to transmit.

The IEEE 802.3 CSMA / CD standard sends data in variable size frames commonly called *packets* with a minimum spacing of 9.6 microseconds.

For any hub, if there is activity (signal) on more than one input, a collision is assumed. A special signal called the *collision presence* signal is generated. This signal is generated and sent out as long as activity is still sensed on any of the input lines after a collision is detected. This signal is interpreted by every node as an occurrence of collision.

4.4 ASC To LSC Communication Protocol :

The ASCs will be used to monitor and control the shop floor test equipment and the automated workcells.

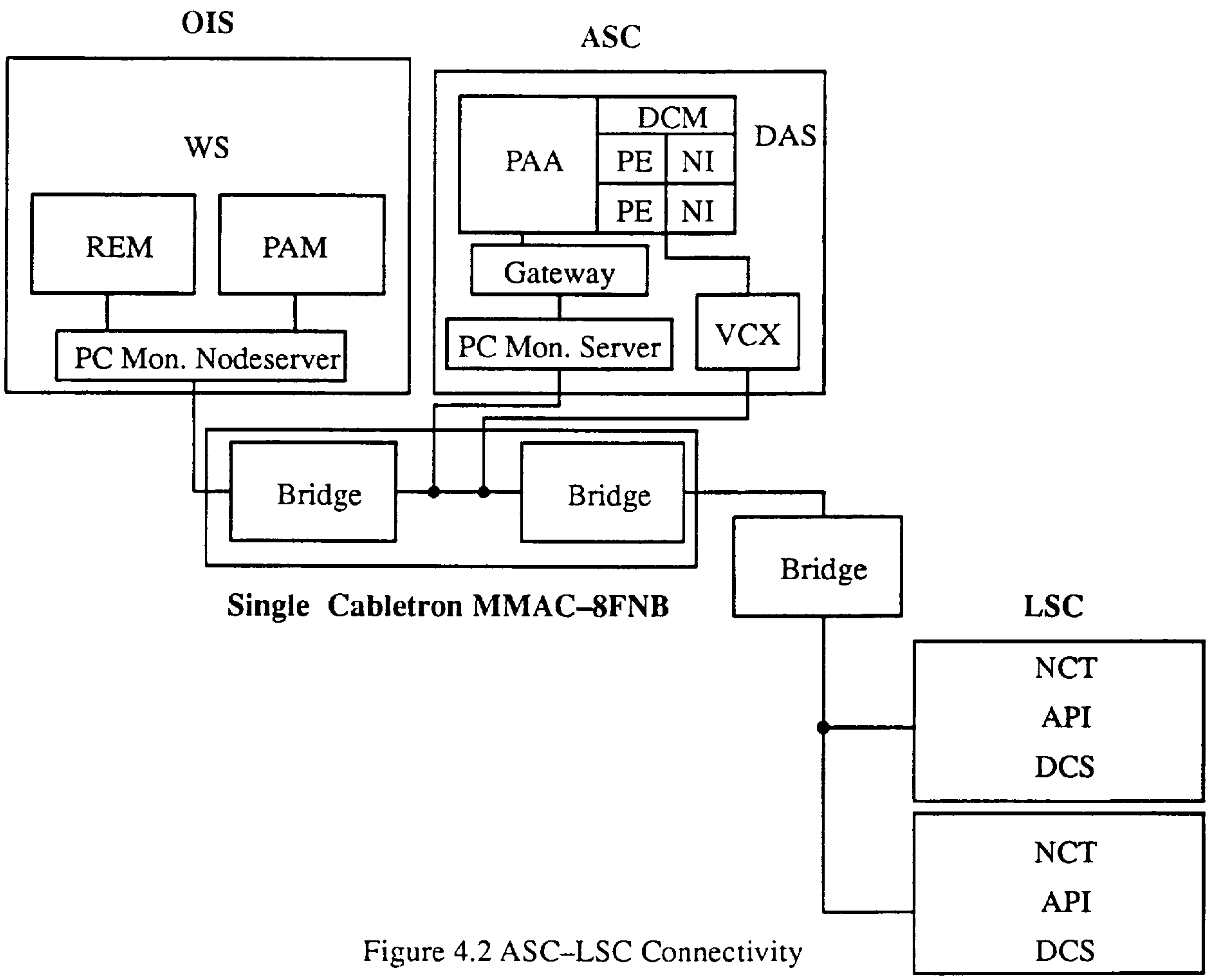


Figure 4.2 ASC–LSC Connectivity

At the Local Supervisory Computer (LSC) level, data from the workcell devices are collected by utilizing software provided by RUST International. Data collected at the LSC is transferred to the ASC by a combination of BASEstar request/response transactions and LSC initiated File Transfer

Protocol (FTP). After the data has been successfully transferred to the ASC, the ASC will process the data and pass it to the Work Stream (WS) product on the OIS system. After reaching the OIS, the data is made available for analysis and manipulation by the other software.

The vendor Device Control Software (DCS) may request Network Cooperating Task (NCT) services by utilizing the Application Program Interface (API). The NCT and API communicate through sockets using the Transport Control Protocol (TCP) protocol. The API acts as a client and the NCT provides services to the API. In other words, the execution of an API routine generates a request that is transmitted to the NCT. Consequently, the NCT generates a response that returns the requested information to the API routine that initiated the transaction. In addition to the API interface, the NCT functions as a server for the network Device Access Software (DAS). The BASEstar product resides on the ASC, and provides an API. Hence, the Process Automation Application (PAA) can invoke NCT functions by including BASEstar API calls in the PAA source code.The BASEstar/PAA communicates with the NCT in the same manner as the API. The BASEstar/PAA formulates requests for NCT services and sends the requests to the NCT. Upon receipt of a request, the NCT validates the request and formulates the appropriate response. Next, the NCT's response is returned to the requester. Finally, the PAA sends the acquired information to the WS by utilizing the WS BSGW interface.

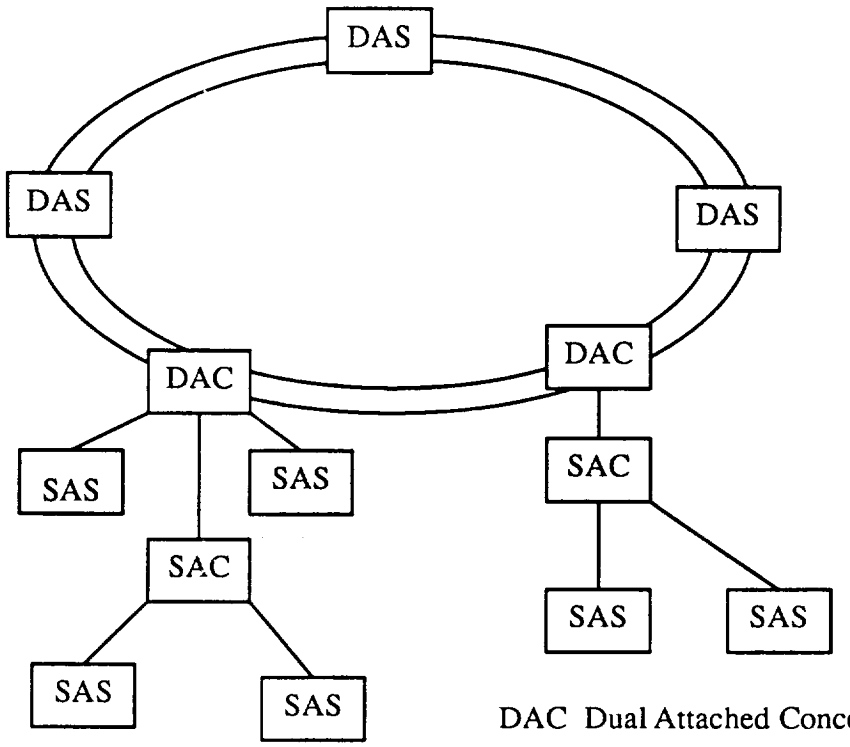
In short, the process can explained as the data is passed from the DCS to the NCT by incorporating API calls into the DCS source code. Data from the NCT is passed to the ASC by adding the BASEstar API calls into the PAA source code. The PAA processes the acquired data and passes the data to the WS through the WS/ BASEstar Gateway (BSGW).

4.5 FDDI Dual Ring Of Trees [11] :

A typical FDDI network consists of the following four types of nodes : Dual Attached Stations (DAS), Dual Attached Concentrators (DAC), Single Attached Stations (SAS), and Single Attached Concentrators (SAC).

A preferred FDDI topology consists of a dual ring of trees. The trunk ring is formed with dual attached stations and concentrators. The trees are formed of concentrator connections to SAS and SAC. Typically the dual ring itself would consists of concentrators, bridges, routers, file servers, main frame computers, etc. Workstations and other desktop computers would be connected through concentrators to form trees.

The use of concentrators to form tree structures offers a number of advantages. It allows lower cost SASs to be connected to the ring. It enhances network reliability since a concentrator automatically reconfigures the network as stations are inserted or re-inserted from the tree. It also rejects links that are faulty and ensures that they do not bring down the ring. A concentrator also allows the use of a star wiring topology that is present in most buildings.



DAC	Dual Attached Concentrator
DAS	Dual Attached Stations
SAC	Single Attached Concentrator
SAS	Single Attached Station

Figure 4.3 FDDI Dual Ring Of Trees

4.6 One Logical FDDI Ring At The ASRM Site :

A simplified figure of the network structure at the ASRM will be as shown in Figure 4.4

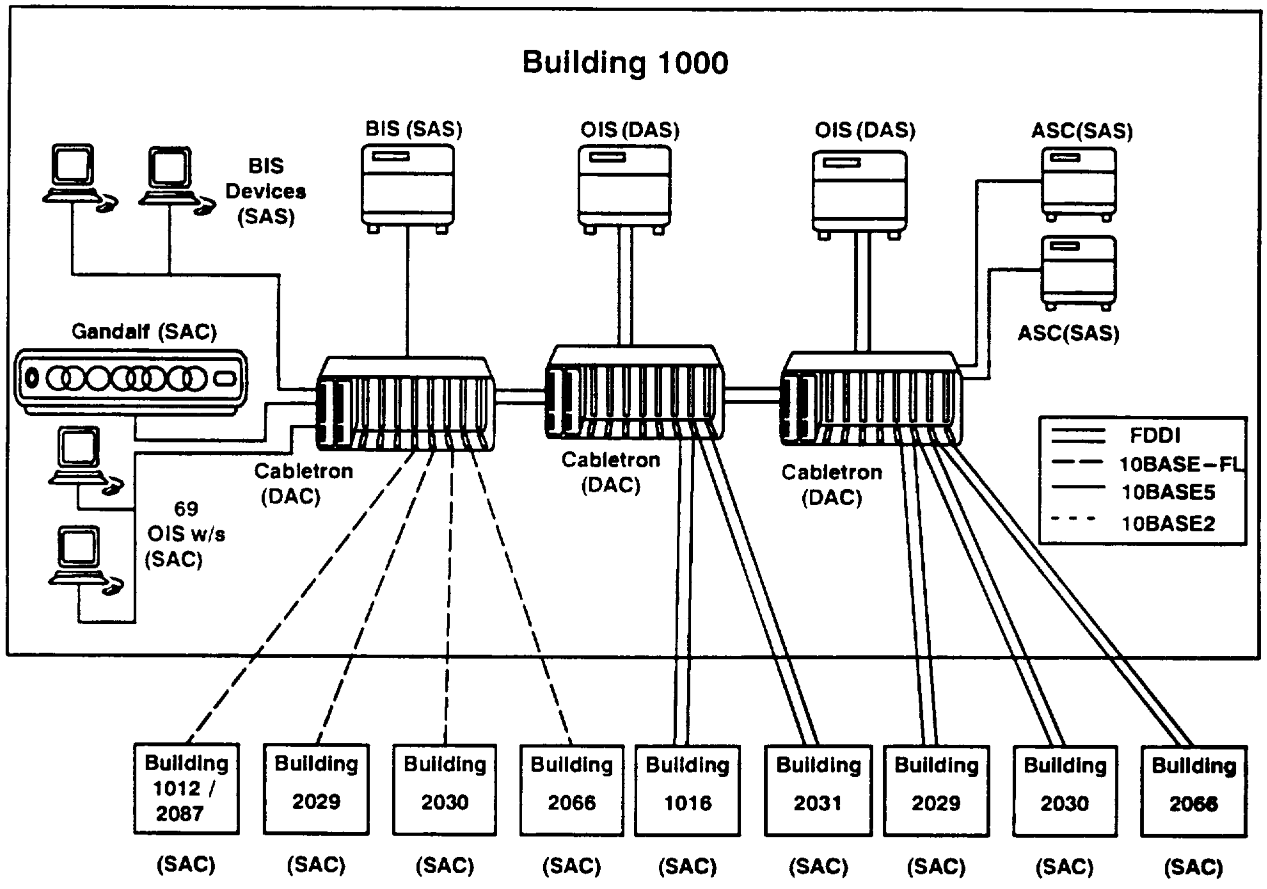


Figure 4.4 FDDI Ring At The ASRM Site

Comparing Figure 4.4 and 4.3 shows that logically a dual ring of trees exist at the ASRM site. So though physically it is a point–to–point connection, the overall network structure will have one logical FDDI ring acting as a backbone for the whole complex.

5. CONCLUSION

5.1 Network Expectations :

The network should be reliable and have redundant links since

1. The control of the manufacturing will be accomplished over the LAN via APIs.

2. Large amount of data will be required due to the critical nature of the solid rocket motors in the Space Shuttle program.

5.2 Network Evaluation Parameters :

The two primary evaluation parameters that will be used to judge the network performance are *Throughput* and *Delay.*

The Throughput is the effective bit rate of the system in bits per second (bps). It does not include the overhead bits used by the protocol or the packets that have to be transmitted again.

The Delay in a LAN is judged by the mean delay per packet. This mean delay per packet will be used to calculate the time to transmit a graphics page.

The delay in a LAN is contributed mostly by the following three factors

1. Propagation Delay
2. Delay in a Transceiver
3. Queue Delay in a Bridge

Also the user response time will be important.

5.2.1 Propagation Delay :

The propagation delay of the light signal traveling down the fiber will be modeled by using a *fixed delay* model provided by BONeS. The link delay will be calculated by dividing the distance by the speed at which the light travels down the fiber (0.67 times the speed of light). The link distances and the corresponding delays for each link are shown in Table 5.1.

Link	Distance (feet)	Number of Workstations on the link.	Number of Workcells on the link.	Type of the Link	Link Delay (microsec.)
Link # 1 : (2029)	6700	29	04	Critical and Non–critical	10.16
Link # 2 : (2031)	4650	15	00	Critical	7.05
Link # 3 : (1016)	1450	27	16	Critical	2.20
Link # 4 : (1000)	00	19	00	Non–critical	0.00
Link # 5 : (2066)	3550	09	01	Critical and Non–critical	5.38
Link # 6 : (2030)	5000	12	06	Critical and Non–critical	7.58
Link # 7 : (1012)	950	14	00	Non–Critical	1.44

Table 5.1 Propagation Delay Table

5.2.2 Delay In A Transceiver :

The worst case collision detection time of a particular CSMA / CD commercial transceiver was found to be 900 nanoseconds [15–16]. The worst case packet delay of a particular commercial optical hub was found to be 630 nanoseconds [17]. This delay is caused by the light–to–electrical and electrical–to–light conversion.

5.2.3 Queue Delay In A Bridge :

In simulation each of the CSMA / CD networks will be connected to the FDDI backbone by a bridge. A bridge converts the CSMA / CD packet to an FDDI packet. It buffers the incoming packets until they are serviced. If a packet enters the bridge and the queue is full, the packet is rejected. If the queue is big but not full, then the packet will not be rejected but it will be delayed.

5.3 Network Modeling Using BONeS :

The commercial software package Block Oriented Network Simulator (BONeS) will be used for simulating the network.

BONeS is an event driven simulation. Each event has to be triggered by a previous event called a *Trigger*. If a block is not triggered then there will be no output. Thus when building model using the provided blocks, race conditions such as parallel inputs must be avoided. Instead, blocks should be cascaded to prevent the race conditions.

Different nodes that will be modeled are

1. Workstations
2. Workcells

Models of CSMA / CD nodes, FDDI nodes, and bridges are included in the BONeS library.

The parameters of the CSMA / CD nodes will be set to the IEEE 802.3 CSMA / CD standard. But the packet size will be set to the smallest possible size of 64 bytes for a worst case analysis. If the packet size is small the transmission time will be small, with respect to the propagation delay, and more collisions will occur.

A traffic source model will be developed to model a workstation sending a block of data such as a text or graphics screen. The traffic source model sends a set number of packets at an interarrival rate set by the user. The interarrival rate has a Poisson distribution because traffic on a LAN tends to have a Poisson distribution.

The Table 5.2 gives the list of BONeS modules that will be used for each device at the ASRM site.

Name of the device at the ASRM site	Type of the device	BONeS Module that will be used
OIS	FDDI node	FDDI node
BIS	CSMA / CD node	CSMA / CD node
ASCs	CSMA / CD node	CSMA / CD node
FDDI workstations	FDDI workstation	FDDI workstation
CSMA / CD workstations	CSMA / CD workstation	CSMA / CD workstation
Workcells	CSMA / CD workstation	CSMA / CD workstation
Cabletron EMME card	Ethernet bridge	CSMA / CD Bridge
Cabletron FDMMIM	FDDI to Ethernet bridge	FDDI to CSMA / CD Bridge
Cabletron MT8–MIM	DELNI card	Fixed Delay
Cabletron FORMIM	10BASE–FL card	CSMA / CD hub
Cabletron TPRMIM	10BASE–T card	CSMA / CD hub
Cabletron CXRMIM	DEMPR card	CSMA / CD hub
Cabletron GX–M	GatorStar card	LocalTalk to CSMA / CD router.
Cabletron FDMIM–04	FDDI concentrator	FDDI hub

Table 5.2 BONeS Modules to be used

5.4 Data Rates For The Workstations and The Workcells

5.4.1 Data Rate For The Workstations :

The data rate for the workstations can be computed by assuming that the workstation will be sending a block of data such as a text or graphics screen. A page of graphics is assumed to be 640 pixels by 480 lines with 16 colors. The data to be transmitted is given as 1920 characters / screen [4] which is equal to 192 kilobytes of data. The number of packets required to send 192 kilobytes can be calculated using 64 byte packets (worst case design). The delay per graphics page can be calculated by multiplying the number of packets by the mean delay per packet.

5.4.2 Data Rates For The Workcells :

The data rates for the workcells are computed using the information obtained [2] and is tabulated in Table 5.3 below.

Different types of workcells and their operations can be classified as follows.

1. Development type.

a. File transfer

b. File delete

2. Manufacturing type.

a. File transfer

b. Write

c. Read

d. Polling

e. File delete

3. Testing type.

a. File transfer

b. Write

c. Read

d. File delete

4. Non–Destructive type.

a. File transfer

b. Write

c. Read

d. File delete

Building No.	Workcell ID #	Description	Workcell Type	Data Rates	Duration
B_1016	W 102	Robotic washout station	MFG	441.6 bytes/ sec	24 hrs
	W 104a	Hydrotest equipment	Testing	15.66Mbytes /sec	8 min/ day
	W 104b	Hydrotest data acquisition system	Testing	15.66Mbytes /sec	8 min/ day
	W 107	Electromag–acoustic eddy current test	NDE	533 bytes/ day	16hrs / 3 days
	W 114	Robotic dimensional inspection	NDE	1.44 Mbytes /day	12 hrs/ 3 days
	W 116	Aqueous degreaser	MFG	384 bytes/ sec	8 hrs
	W 117	Robot clean/paint/osee	MFG	364.8 bytes/ sec	40 hrs/ 5 days
	W 118	Plastic media blast robot	MFG	105.6 bytes/ sec	16 hrs/ 5 days
	W 121	Clean, dry, liner robot	MFG	604.8 bytes/ sec	16 hrs/day
	W 148	Horizontal elastic insulation application	MFG	268.8 bytes/ sec	60 hrs/ 5 days
	W 149	Pattern cutting station	MFG	720 bytes/ sec	4 hrs/day
	W 159	Aqueous degreaser	MFG	384 bytes/ sec	8 hrs
	W 160	Plastic media blast robot	MFG	652.8 bytes/ sec	8 hrs/ 5 days
	W 161	Component Washout Robot	MFG	182.4 bytes/ sec	8 hrs/ 5 days
	W 168	Ultrasonic inspection	NDE	533 bytes/ day	16 hrs/ 3 days
	W 169	Autoclave (insulation curing)	MFG	86.4 bytes/ sec	40 hrs/ 5 days
B_ 2029	DCS	Mix/Cast distributed control system	MFG	960 bytes/ sec	96 hrs/ 1 month
B_2030	W 402	Real time radiography	NDE	800 bytes/ hr	40 hrs/ 3 days
	W 403	Ultrasonic test	NDE	800 bytes/ hr	24 hrs/ 3 days
B_2060	WSSP	Small scale propellant		768 bytes/ sec	constant
	??	Scales		18816 bytes /day	constant

Building No.	Workcell ID #	Description	Workcell Type	Data Rates	Duration
B_2066	LIMS	LIMS		23.81 bytes/ sec	3–4 min
B_2076	MTRQ	Motor qualification data acquisition system		576 Kbytes/ sec	8–12 min/ day
B_3003	??	Propellant removal station	MFG	441.6 bytes/ sec	24 hrs
B_3005	??	Thermal treatment control	MFG	384 bytes/ sec	8 hrs
B_3010	??	Incinerator	MFG	384 bytes/ sec	8 hrs
B_3011	??	Feeder preparation	MFG	384 bytes/ sec	8 hrs

Table 5.3 Workcells and their data rates

5.5 Preliminary Conclusions :

1. The FDDI backbone should cause only a negligible delay compared to the other delay–causing factors considered in section 5.2.

2. The FDDI ring bandwidth should be sufficient for the whole network. However there is possibility of overload as all the traffic is directed towards B_1000.

3. The throughput should linearly increase with the traffic intensity per node.

4. The optical hubs should be arranged to cascade as few as possible to avoid large propagation delays. One commercially available optical hub specifies that each additional hub in a path reduces that path by 180 meters (590 feet).

APPENDIX

A1. BONeS SIMULATOR

The Block Oriented Network Simulator (BONeS) provides an interactive graphical environment for simulation–based analysis and design of a broad range of communication networks. The integrated BONeS environment includes the capability to :

1. Graphically describe data structures in a hierarchical fashion.

2. Graphically describe protocol functions, node processing, and network topology in a hierarchical fashion using block diagrams.

3. Translate the network model into a C program, and execute an event driven simulation of the model.

4. Perform design iterations and tradeoff analysis.

5. Document both models and results.

BONeS provides an easy–to–use modeling and simulation environment, an excellent model library that is user extensible, and a set of powerful analysis tools. BONeS minimizes the amount of code the user has to write and provides on–line help, documentation aids, and error checking. These features free the user from the low level details of simulation programming and directs the focus on modeling, analysis, and design.

In BONeS environment, the network model is specified in terms of the *network topology, traffic, packet and message (data) structures,* and *protocol functions.* The user constructs the network graphically and hierarchically using the building blocks from the BONeS model library. The user can also write the components of a model in C and incorporate them into the BOneS modeling environment. BONeS translates the network model into a C program, executes an event driven *Monte Carlo simulation*, computes statistical measures of network performance, and displays the results graphically.

APPENDIX

A2. CABLETRON DEVICES

The communication network structure of the ASRM will be using different Cabletron devices and modules for internetworking.

This part of the report presents a summary of the Cabletron devices to be used.

A2.1 Cabletron's Multi Media Access Centers (MMACs) :

The Cabletron MMAC is an advanced network intelligent wiring hub that integrates LANs and WANs, in Ethernet, Token Ring, and FDDI environments.

The MMAC's architecture also provides media independence, allowing selection of unshielded twisted pair, shielded twisted pair, thin or thick coaxial cable, optical fiber, and AUI transceiver cabling.

The MMACs provides a backplane referred to as the Flexible Network Bus (FNB) to interconnect different modules internally.

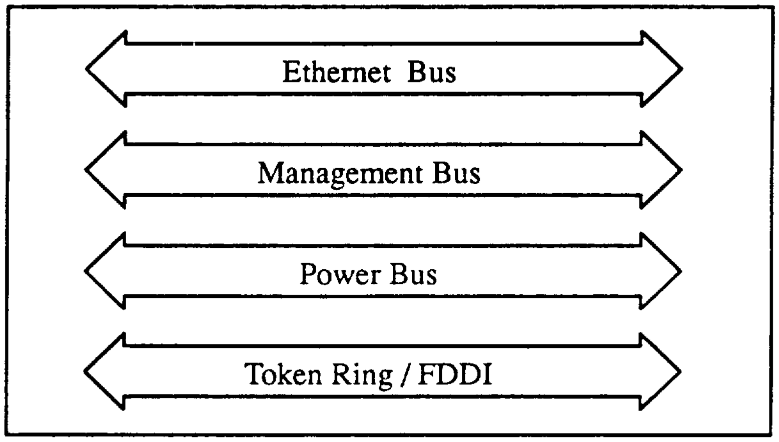


Figure A2.1 MMAC Strucutre

The catalog does not specify the bandwidth for the FNB but refers to it as *a large highway of many lanes*.

The Cabletron devices provide network redundancy in two forms. The first method is to ensure that all data connections have two back–up paths. This method allows critical servers, nodes, or backbones to be backed–up with multiple data paths from one or more MMACs. In event of a data path failure, back–up paths take over. This feature will be useful in connecting the manufacturing critical hubs to B_1000 at the ASRM site.

The second method of redundancy built into the MMAC is its load sharing / redundant power supplies. This method will be used for sharing the load of one of the OIS VAXEN cluster by the other in the event of heavy loading of one of the OIS VAXEN.

The MMACs chassis are modular. allowing one to *hot swap* media boards and power supplies. This feature reduces the downtime, as units can be serviced quickly without special tools.

The MMACs allows for connection of up to seven Media Interface Modules (MIM) when utilizing MMAC–8FNB, up to four MIMs for MMAC–5FNB, or two MIMs for MMAC–3FNB. The first card slot in all the MMAC devices must contain one of the Cabletron's intelligent repeater or management modules. The other slots within the MMAC can be devoted to a variety of Cabletron MIMs.

At the ASRM site all the MMAC devices will be MMAC–8FNB and the first slot in the MMAC will be the EMME multichannel management / bridge module.

A2.2 Cabletron's Media Interface Modules (MIMs) :

A MIM is an add–on card that can be plugged into a slot of a MMAC. Each MIM has ports on its faceplate to support cabling. There are over 50 different types of MIMs. Each MIM functions uniquely as a repeater or as a bridge. A single MMAC device can have different types of MIM (Ethernet, Token Ring, and FDDI), each performing different function. A bridge module is required when different MIMs are used in a MMAC.

The different MIMs to be used in the network at the ASRM are listed in Table A2.1.

Name of the Card	Type of the Card	Protocol	Number of ports	Comments
EMME	Ethernet Bridge	Ethernet	4 ports	Used as a management module in all the MMACS
FDMMIM	FDDI to Ethernet Bridge	—	8 ports	Connects 10 Mbps Ethernet to 100 Mbps FDDI
MT8–MIM	DELNI Card	—	8 ports	AUI Transceiver
FORMIM	10BASE–FL Card	Ethernet	12 ports	Provides connectivity for 12 Ethernet channels
TPRMIM	10BASE–T Card	Ethernet	24 ports	Provides Connectivity for 24 Ethernet Channels
CXRMIM	DEMPR Card	Ethernet	12 ports	Provides Connectivity for 12 Ethernet Channels
GX–M	GatorStar Card	LocalTalk	24 ports	Integrates 24 port LocalTalk repeater with a LocalTalk to Ethernet router
FDMMIM–04	FDDI Concentrator	FDDI	4 ports	Provides 4 concentrator ports for FDDI connections

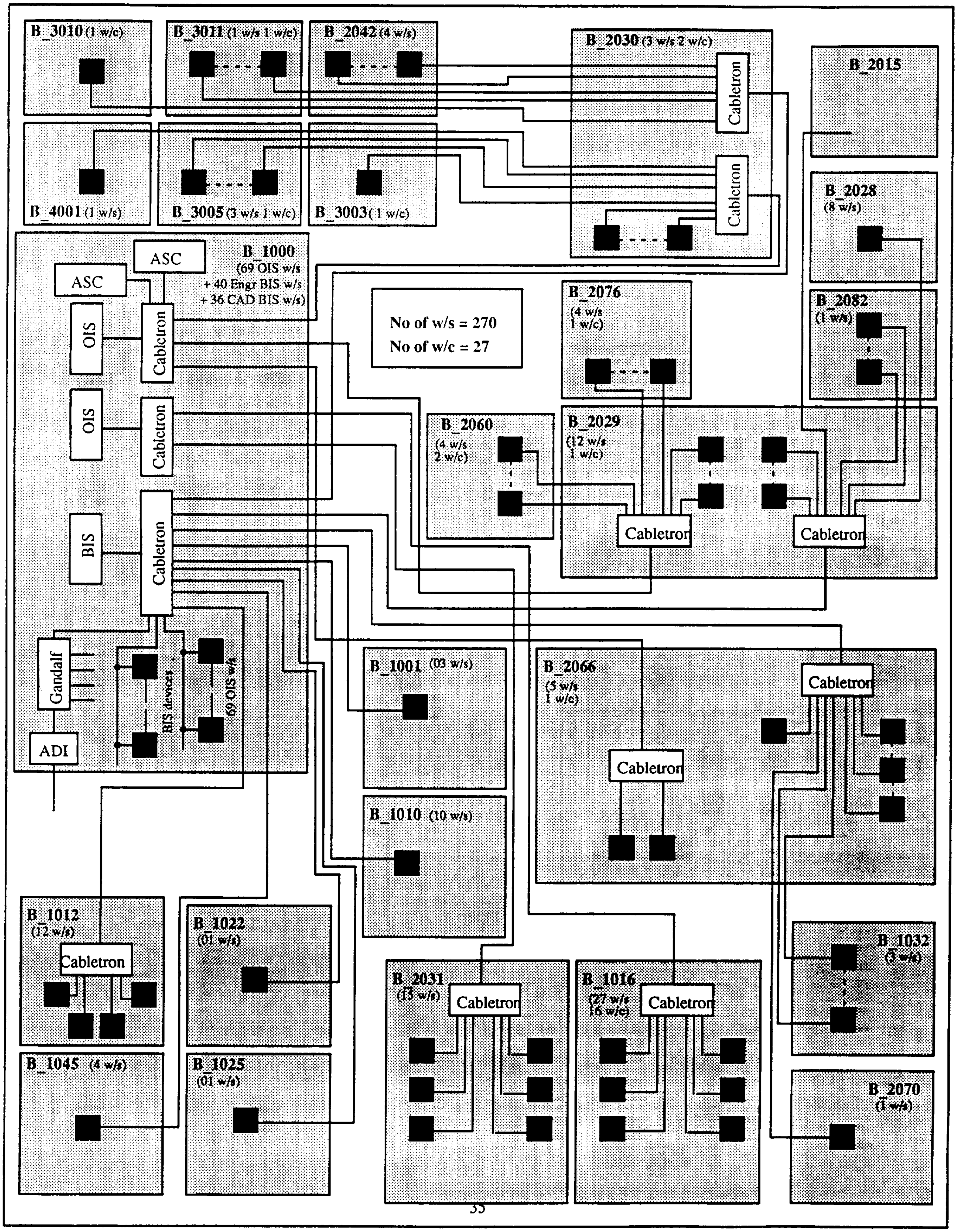
Table A2.1 Cabletron MIMs used at ASRM

A2.3 Summary :

A Cabletron device is a standard computer sized cabinet with 4–8 slots. Each card is bridged to each other and is hot swappable. Reconfiguring a network logically and physically can be done easily with the Cabletron device. The Cabletron device automatically configures to the fastest path. It can be fine–tuned to increase the performance or throughout of a network.

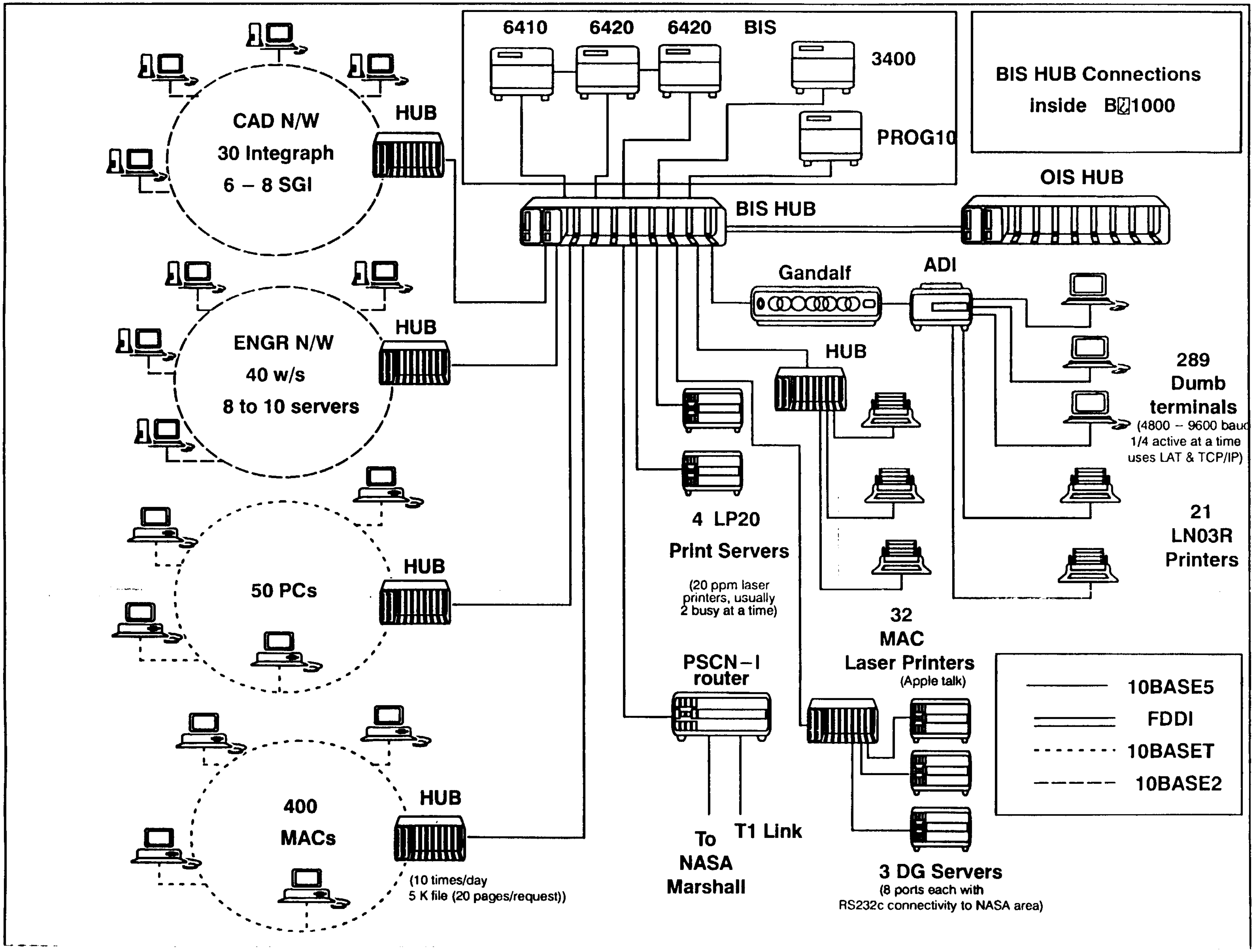
APPENDIX

A3. ASRM NETWORK STRUCTURE


B_3010 (1 w/c)
B_3011 (1 w/s 1 w/c)
B_2042 (4 w/s)
B_2030 (3 w/s 2 w/c)
Cabletron
Cabletron
B_2015
B_4001 (1 w/s)
B_3005 (3 w/s 1 w/c)
B_3003 (1 w/c)
B_2028 (8 w/s)
B_1000 (69 OIS w/s + 40 Engr BIS w/s + 36 CAD BIS w/s)
ASC
ASC
OIS
Cabletron
OIS
Cabletron
BIS
Cabletron
Gandalf
ADI
BIS devices
69 OIS w/s
No of w/s = 270
No of w/c = 27
B_2076 (4 w/s 1 w/c)
B_2082 (1 w/s)
B_2060 (4 w/s 2 w/c)
B_2029 (12 w/s 1 w/c)
Cabletron
Cabletron
B_1001 (03 w/s)
B_2066 (5 w/s 1 w/c)
Cabletron
Cabletron
B_1010 (10 w/s)
B_1012 (12 w/s)
Cabletron
B_1022 (01 w/s)
B_2031 (15 w/s)
Cabletron
B_1016 (27 w/s 16 w/c)
Cabletron
B_1032 (3 w/s)
B_1045 (4 w/s)
B_1025 (01 w/s)
B_2070 (1 w/s)

APPENDIX

A4. BIS HUB Connections inside B_1000


BIS HUB Connections inside B[illegible]1000
6410
6420
6420
BIS
3400
PROG10
OIS HUB
BIS HUB
CAD N/W
30 Integraph
6 – 8 SGI
HUB
ENGR N/W
40 w/s
8 to 10 servers
HUB
50 PCs
HUB
400
MACs
HUB
(10 times/day
5 K file (20 pages/request))
Gandalf
ADI
HUB
289
Dumb
terminals
(4800 – 9600 baud
1/4 active at a time
uses LAT & TCP/IP)
21
LN03R
Printers
4 LP20
Print Servers
(20 ppm laser
printers, usually
2 busy at a time)
32
MAC
Laser Printers
(Apple talk)
PSCN–I
router
To
NASA
Marshall
T1 Link
3 DG Servers
(8 ports each with
RS232c connectivity to NASA area)
10BASE5
FDDI
10BASET
10BASE2

APPENDIX
A4. X WINDOW TRAFFIC

Introduction

About X Window system

The X window system, or just X for short, is a combination of several software components working together to provide a high performance graphical interface to users. In the X environment, users do their jobs on network–based and bit–mapped windows that are controlled and managed by a terminal server. X supports multiple windows so that it provides users with a multitask application environment. Users may open a window for each interactive applications. The X components are many things. From the user's point of view, they include a window server, a window manager, a display manager (or none), and a collection of application programs (called clients). To the X programmer, they also contain a communication protocol and a structured library that is the lowest level routines with access to the X protocol. Programmers use this library to build user interfaces.

The X window system is well described by a *client–server* model with a communication protocol between the client and the server. The *server* is a software program residing at the terminal. It takes user inputs (from the keyboard, mouse, etc.) and sends them to relevant programs (*clients*). Then it receives the client display requests and does the actual drawing on the screen. The *client* is the application program the user runs. It may run locally if the terminal is a workstation, or run remotely at a mainframe with results being displayed at simple terminals. The X protocol controls the various client–server interactions supported by X. It is defined as a set of primitives: *request, event, reply,* and *error.* The request is an order sent from a client to the server as a response to a user input/action to request the server to make some changes to the graphics display, e.g. creating/closing a window. The client may also send a request to the server for some information from the server. In this case, no drawing action will be taken. The event, sent from the server to a client, is used to indicate a device action has been initiated by the user. The difference from conventional client–server relationships is that the server gets inputs from users then produces relevant events to inform clients. A reply is used by the server to send some information a client requested. A client may request and retain the server information when being created so that the amount of interaction traffic is reduced. The server sends error messages to the client to report errors.

History, Standards, and Application Development

X was originally developed at the Massachusetts Institute of Technology with the help of various manufacturers in 1984. The first commercial implementation of X was introduced by DEC, running on the VAXstation under the Ultrix operating system. The latest version of X is the Version 11 (X11). It is a joint effort by major computer vendors. The first release of X11 (named X11R1) became available in 1987. Then came the MIT X Consortium, a large and influential collection of software and hardware companies, to keep the X open to all. The X Consortium conducts the current development of the X window system. The latest version is X11R5, released in Dec. 1991. However, X11R4 is still the most widely used. The X window system has been widely accepted as a *de facto* standard for distributed window management.

The X11 standard consists of a C programming library Xlib and an inter–client communication protocol (often called the X11 protocol). The Xlib is a large collection of C language routines which provide a lot of basic windowing functionality. It is the lowest programming interface to the underlying X protocol. Since X was designed to be "policy free", it does not specify a standard user interface. It is up to the X programmer to write his/her favorite user interface. X allows any style of user interface by providing a very flexible set of routines in Xlib. The X11 protocol is the core of the X window standard. All the communications between the client and server are through this protocol. To implement a X system is to implement the protocol. It is a bidirectional asynchronous stream–oriented protocol. It may operate on top of any network layer as long as the underlying network layer provides a sequenced and unduplicated byte delivery service. If the client and server are on the same machine (workstation style), the protocol is typically supported by an interprocess communication mechanism. Otherwise, a network connection is assumed. With asynchronous communication, there is no explicit acknowledgement by the client/server. The client/server process does not have to wait for the consequences of previous packets, but assumes reliable receipt and processing. This approach speeds up the system operation and improves the network performance. Another improvement to the network efficiency of the X protocol is its ability to group a number of requests into a single packet before sending them over the lower network layer. This can greatly reduce the network overhead.

The X11 standard is the heart of the X window system. It was designed to be terminal device independent and network transparent. X developers are encouraged to develop window systems most suited to users interests. They do not have to have the knowledge of the underlying network that is to support the X protocol. The display and the applications can be running on different machines over a network. This distribution ability allows X users to access almost all network resources and to take advantage of different platform capabilities via the network.

The only device dependent part in X system is the server. Running at the terminal (or workstation), the window server takes display requests issued by clients through the X protocol and generates the graphic output on the screen. Platform vendors are free to optimize their server software for their hardware.

Although Xlib is a basic interface for X programmers to build up application programs, it does not provide programming efficiency. To create a simple display object will involve a number of Xlib routines. This can be analog to programming with assembly language. To allow easy application creation, X vendors put extensive effort into the development of toolkits, e.g., MIT 's X Toolkit Intrinsic (also known as Xt Intrinsic) and OSF's Motif. Motif is even built upon the Xt Intrinsic. These toolkits are the high–level languages of the X window system. A general X window system structure is shown in Figure A4.1.

Another thing that needs to be pointed out is that X is intended to be operating system independent. It only requires a reliable data communication service between the server and clients. However, some network protocols reside in the operating system kernel, e.g., TCP/IP in the BSD UNIX operating system and the DECnet protocol with DEC's VAX/VMS machines. Certain factors related to

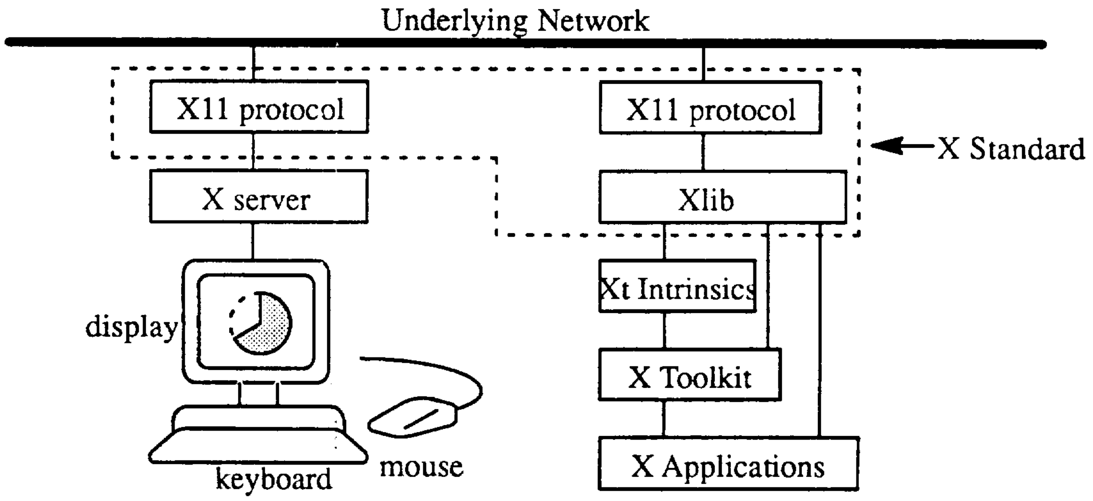


Figure A4.1 A General X Window System Structure

the operating system may need be taken into account for X implementation. But as long as there exists a data communication channel between the server and clients, the server can take the request and produce output for the client no matter what operating system the client is running on.

Some special client programs

There are two special client programs: the window manager and the display manager.

The standard release of the X window manager from MIT is called *twm*. Several other window managers, such as *mwm* (the Motif window manager from OSF), *awm* (Ardent window manager), *rtl* (tilted window manager, developed at Siemens Research and Technology Laboratories, RTL), and *olwm* (the OPENLOOK window manager from AT&T and Sun) are also widely used. The window manager takes care of the layout of the windows. It allows users to move and resize windows without client interactions. The reason for having the X window manager be a client program separated from the X server is to let users be able to choose the window manager they like, so that the design of the graphical user interface is freed from the manufacturer–selected graphics system. Another important functionality of a window manager is to provide a means for inter–client communications. A typical example is the *cut* and *paste* procedure between two windows on a display. The inter–client communication of X is directed by the ICCCM (Inter–Client Communications Conventions Manual).

The display manager is a client program designed to start the X server automatically and to keep it running. The *xdm* is a X display manager released from MIT. It is started by the root, not the user, at the system startup. When first run, *xdm* reads in some configuration parameters from a file named *xdm–config* in the xdm directory. This file indicates which servers *xdm* will manage. After reading the list of servers, *xdm* starts up these servers. For remote servers, *xdm* opens network connections. Then the display manager emulates the *getty* and *login* procedure on the displays, prompting for a user's name and password. When a user enters a name and password, *xdm* checks them and, if correct, executes initialization of a window application, creating the selected clients and a window manager defined in the file named .Xsession in the user's home directory. The file .Xsession is similar to .xinitrc. When the user logs out, *xdm* will destroys all windows by running a cleanup script called .Xreset and then goes back to the login prompt. The big difference when using an X display manager is that the X server is kept alive when the user logs off, waiting for a connection. In fact, when *xdm* is started up at system initialization, it brings up all the servers it manages. This is the typical situation of implementing X windows on X terminals with all clients running on a remote host.

Implementations of X

There are three ways to implement the X window system [20].

The first is to use workstations. The X server and all clients run locally. There is no extra network traffic introduced by X windows. If a distribution of load and high processing speed are needed, workstations are the choice.

The second way is to implement X on PCs running DOS or MS Windows. This requires adding PC X server and network protocol software (e.g. TCP/IP) to PCs. Client programs run remotely on some host server. The advantage of using PCs is to make full use of already installed PCs which still can run DOS or MS Windows applications. The PC hardware needs at least 4M bytes of RAM, an 80386SX or faster processor, and higher resolution graphics monitor/boards. High–performance graphics controller boards based on graphics processors (e.g., Texas Instruments TMS34010/20 products) can alleviate the PC CPU of running the server software.

The third implementation of X is by using X terminals. The strength of X terminals is that they are designed to run X windows. They have powerful processors and high–resolution graphics displays, and they are less expensive than workstations. The X terminals only run X servers and network protocol software. But the centralized host which runs all X applications needs lots of memory and processing power. One X terminal may require up to 1 MIPS of processing power, 1M byte of main memory, and 20M disk space on the host. In addition, each X terminal will produce an X traffic stream and load the underlying network. Therefore extra care should be taken during the design period. Figure A4.2 illustrates the three network implementations.

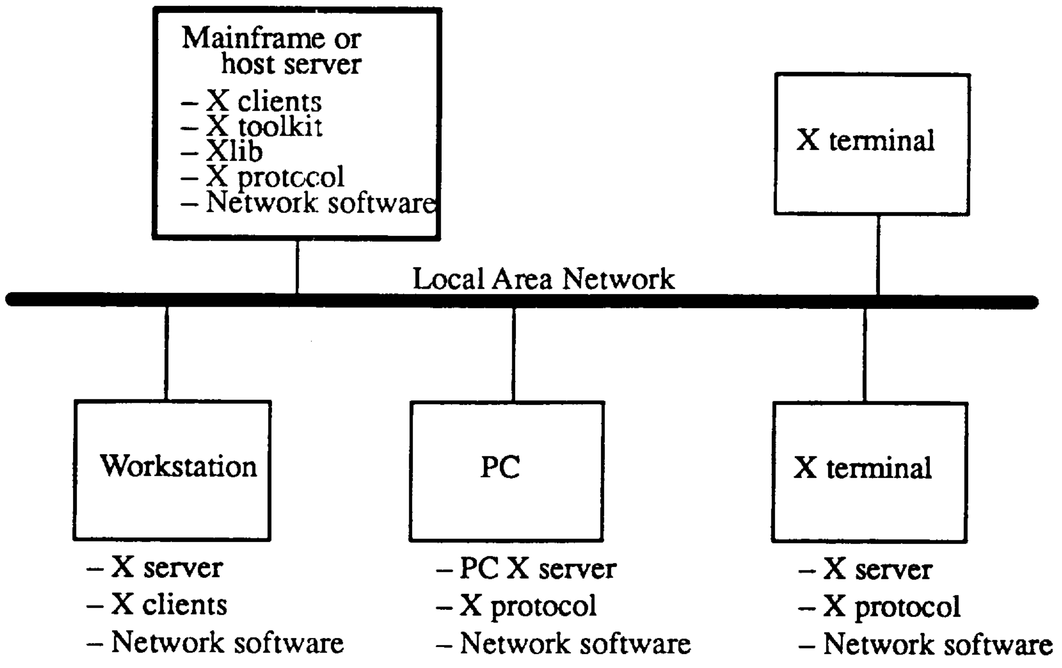


Figure A4.2 Implementations of X over LAN

In summary, the X window system provides a high–resolution, multitasking, graphics interface to end users. At the same time it imposes extra processing power requirements on the terminals and hosts, and data traffic on local area networks.

The following part of this report is divided into two sections. The first section classifies generally the X traffic flows over a local area network for various X implementations. The second section is an investigation of X traffic over TCP/IP and Ethernet.

Traffic Flows: Workstation vs. X Terminal

The X window system is a distributed system. Applications can be run at any node on a network provided that there exists a connection path between the application and the server. On the other hand, X supports a high–resolution, bit–mapped, multi–window interface to the end users. So it will produce extra data traffic compared with the conventional text–based, single screen terminal applications. The X protocol was developed for the data exchanges between the client and server. As previously stated, it has four primitives: *request, reply, event,* and *error.*

The *request* is the only primitive generated by the client and sent to the server. It may carry either an order to the server to draw, to change colors, etc. in a window, or an inquiry for some window information. Every *request* consists of a 4–byte header followed by optional additional data bytes. The maximum length of a request is $2^{18} - 1$ bytes. The server sends a *reply* to the client to reply to an information inquiry contained in a *request.* Each *reply* includes a 4–byte length field followed by none or more additional data bytes. The maximum length of a *reply* is $2^{32} - 1$ bytes. Both the *event* and *error* are 4–byte long, sent by the server. An *event* is used to indicate a device action taken by the user. An *error* is used to report errors.

Under the X protocol is the lower layers of the network protocol function, typically UNIX TCP/IP with Ethernet, or DEC's DECnet protocol. These lower layer protocols put additional overhead on each of X protocol data unit. The X protocol data unit is the data unit delivered as a packet by the underlying protocol. Since the X protocol tends to group several primitives together for transmission to reduce the overhead traffic, one X protocol data unit could comprise more than just one primitive. During a window application, each client will communicate with the server. If clients are distributed at various nodes on a network, there will be a network connection for each of them, as shown in Figure A4.3.

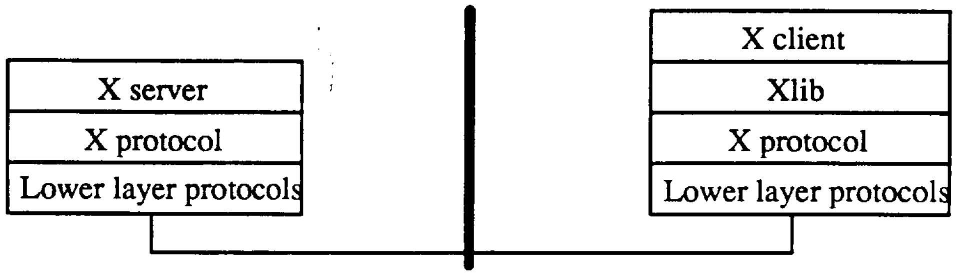


Figure A4.3 X Client–Server Communication Pair

There are several aspects of X traffic composition. The total X traffic over the network is dependent on the distribution of windowing functions. For a workstation version, the traffic may all be transmitted internally by the inter–process communication mechanism. While with an X terminal version, the network must accommodate all client–server data traffic. It is also application dependent. This includes how many terminals are running X windows and what kinds of tasks the users carry out under the X windows. For instance, the end users may be working in working cells equipped with X terminals to interact with a supercomputer running various applications. The end users only execute commands displayed on a selection menu and read the related information in the window. This centralized control does not need much processing power at the terminals, but depends on the speed of the host machine, the host machine interface, and the reliability of the network throughput.

Although X is intended to be network independent and encourages users to develop routines to port the X protocol to the network protocols they want to use, the TCP/IP protocol suite and the DECnet protocol suite are the most widely used. In this report we present an investigation of the actual X protocol traffic and the corresponding data traffic over Ethernet in some special cases. In our study, we assumed a number of identical terminals operating with a centralized host over an Ethernet through X windows and similar traffic statistics, so that the total amount of X protocol traffic on the network is simply the linear combination of traffic of all terminals. The investigation of the actual statistical behavior of aggregated traffic could lead to a further simulation study. We also assumed that each terminal has only one client–server connection active, since the user can only work within one window at a time.

X over Ethernet with TCP/IP

Measurement Configuration

In the study, a client–server connection was made over a 10–Mbits/s Ethernet LAN. Under the X protocol and on top of the Ethernet was the TCP/IP protocol suite. The layered structure is depicted in Figure A4.4 below. Both the machines running the server and client application were Sun SPARCstation 2. The X server was the *X11/NeWS* which supports both the X11 and NeWS protocols. The window manager was the *olwm*, the standard window manager from Sun OpenWindows product. We compared *olwm* with the M.I.T.'s standard window manager *twm*. They showed minimal difference in terms of X protocol data traffic.

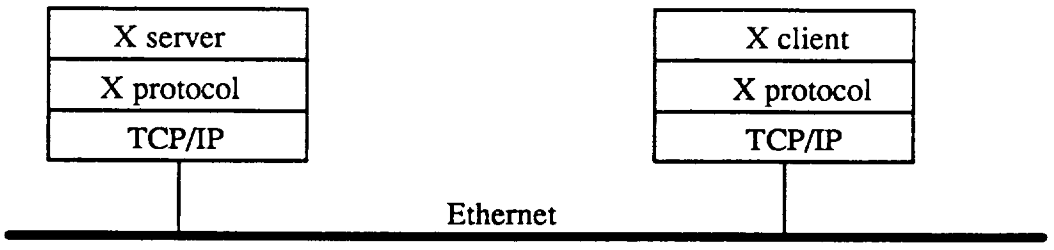


Figure A4.4 Layered Structure

Traffic Monitors

The traffic monitor tools employed included a *perfmeter* and an *xscope*. The *perfmeter* measured the Ethernet packets to and from the server. It recorded the number of packets in each second during the client–server transaction. The *perfmeter* was not able to display the timed characteristics, such as packet delay. But since the measurement took place when the Ethernet LAN was almost idle, the network delay was negligible. The *xscope* utility was a useful program in monitoring X protocol traffic. It sat between the client and server, working as a relay station and at the same time observing and recording all protocol requests, events, error reports, and replies into an output file. Each entry in the output is tagged with a direction, a connection number and a time of day. The data extraction was done by a separate program. *Xscope* is an independent process. It may run on the same machine as the server, or as the client, or on a different host. It makes use of explicit TCP ports to connect the client and server. In our measurement, we setup the *xscope* at the server host and let the *xscope* and *perfmeter* monitor the X traffic in a synchronous manner. Thus the two traffic meters collected the same stream of traffic at two different levels. One displayed the traffic generated directly by the application programs, and the other showed the actual traffic on the network. The statistical differential of the two reflected the overhead of lower layer protocols (TCP/IP and Ethernet).

Application Load

We ran three different client applications.

The first was *xterm*, with which we generated two traffic patterns. One is the interactive UNIX command operations, such as listing directories, changing directories, copying files, cating files, and so on. The other traffic pattern was generated by continuously moving the pointer (mouse) into and out of the *xterm* window (named *xterm2* to distinguish). The second application was running *plaid*, an image pattern generation program. It kept sending drawing requests to the server to display the generated plaid patterns. It was an intensive load we used to investigate the extreme case. The last application was *ileaf5*, a desktop publishing software package from Interleaf, Inc.. This scenario included starting the *ileaf5*, opening a file, page scrolling in the file, closing the file, and closing *ileaf5*.

Measured Data

The data measured with the *xscope* and *perfmeter* are given in Tables A4.1 – A4.3. All entries are measured in bytes unless specially indicated.

Table A4.1 overall statistics

applications	*xterm*	*xterm2*	*plaid*	*ileaf5*
run time (sec.)	251.23	149.56	148.04	239.61
total bytes	1148533	51018	4490780	820129
total units	23919	1748	48822	17097
total X blocks	1237	1311	2287	2175
total pkts	5420	3342	27580	6569
mean pkts/sec.	21.11	21.58	179.12	25.67
max pkts/sec.	72	42	298	103

Table A4.2 clients statistics

applications	*xterm*	*xterm2*	*plaid*	*ileaf5*
total bytes sent	1130173	19010	4489776	755345
percentage (%)	98.40	37.26	99.98	92.10
total units sent	23503	779	48814	15232
percentage (%)	98.26	44.57	99.98	89.19
total blocks sent	835	471	2280	1105
percentage (%)	67.50	35.93	99.69	50.81
block size				
mean	1353.5	40.36	1969.2	683.57
std. dev.	928.09	32.19	260.77	855.91
maximum	2048	404	2048	2048
minimum	4	12	12	4

Table A4.3 server statistics

applications	*xterm*	*xterm2*	*plaid*	*ileaf5*
total bytes sent	18360	32008	1004	64784
percentage (%)	1.60	62.74	0.02	7.90
total units sent	416	969	8	1847
percentage (%)	1.74	55.43	0.02	10.81
total blocks sent	402	840	7	1070
percentage (%)	32.50	64.07	0.31	49.19

block size				
mean	45.67	38.11	143.43	60.55
std. dev.	143.35	23.63	152.94	84.00
maximum	2.32	476	476	2032
minimum	32	32	32	32

The units are the four X protocol primitives: request, reply, event, and error report. The X block (or just block) is defined as the data block the X protocol passes to the lower layer protocol (TCP in our case) for delivery. Since the X protocol works in an asynchronous manner and tries to send units in aggregation, one X data block may contains several units.

Data Analysis

The traffic between the client and the server is highly skewed. Most of the data flows from the client to the server, except in the extreme case, like *xterm2*, where the user kept generating inputs to the server. Even so, the data flow from the server is less than 40 percent of the total.

The X protocol sends its data to the TCP/IP in blocks. Each block contains several data units. Different policies are reflected here. At the client side, one block consists of many data units (an average of 28.15, 21.41, and 13.78 for *xterm, plaid,* and *ileaf5*, respectively), so that a highly efficient use of the underlying network is assumed. At the server side, however, one block only carries one data unit or two. That is because on one hand, the data units sent by the server are mostly *events* whose sizes are very small so that grouping does improve the efficiency, but on the other hand, short blocks may help reduce the response delay, so that user input can be sent to the client application quickly.

The block size depends largely on client applications. For the *xterm2* case, most of the data was generated by the user moving the mouse pointer in and out of the client display window. So the X server kept sending the events EnterNotify, LeaveNotify, FocusOut, FocusIn. And the client responded with short requests specifying the (x,y) coordinates and the GC value (Graphic Context). While in the case of plaid drawing, the client program continued computing the plaid pattern and sending the corresponding drawing requests (PolyFillRectangle) to the server to update the plaid display. If taking the *xterm2* and plaid as two extreme case data, the average block size of *requests* would be around several hundred to one thousand bytes per X block, indicated by the other two cases (1353.5 bytes for *xterm*, and 683.57 bytes for *ileaf*). But the large standard deviations indicate that the block sizes depart the means quite a lot, except in the *plaid* case. These can be visually illustrated by the distribution histograms in Figure A4.5. Most of the time, the block sizes are significantly shorter than the averages. With regards to the network delay performance, the shorter is better for the Ethernet technology.

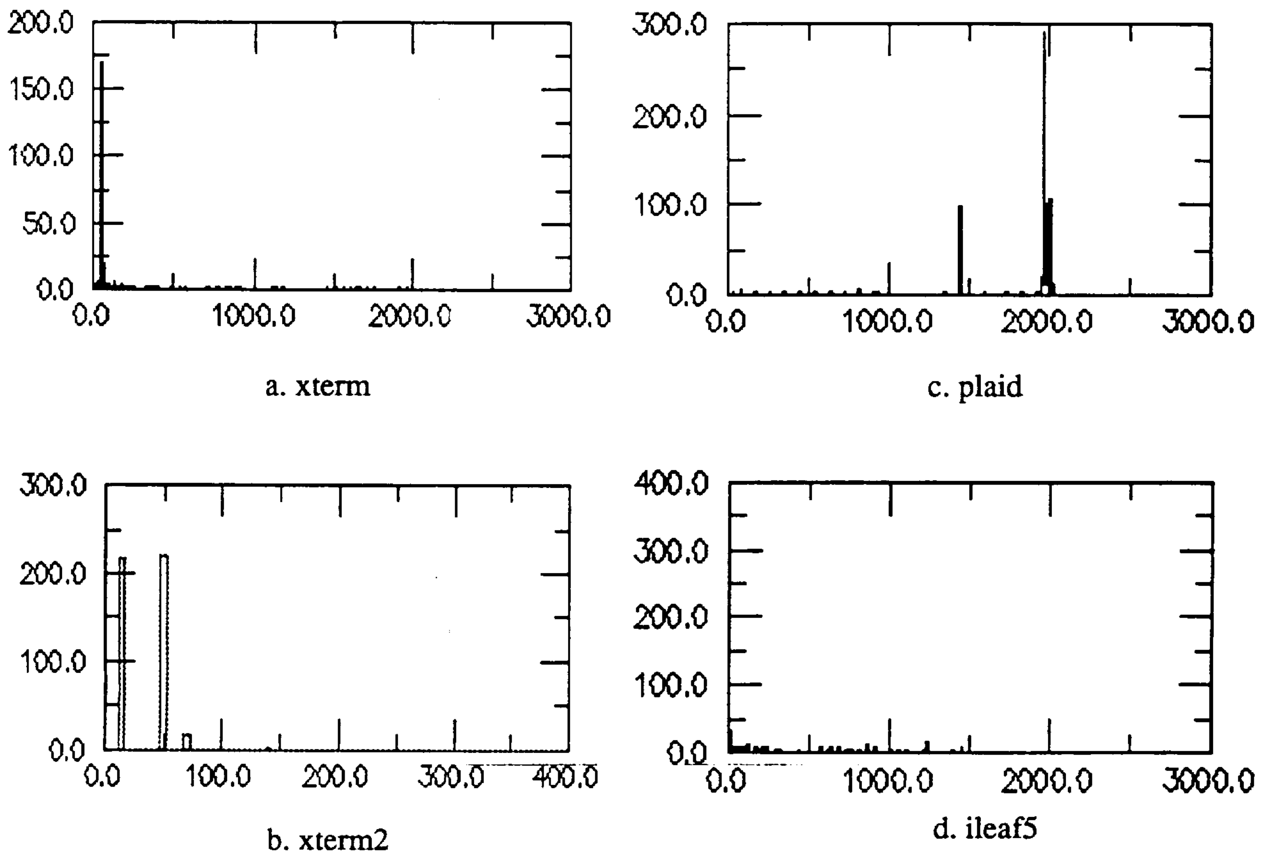


Figure A4.5 Block Size Distributions

The data block sizes from the server are much less than from the client, implying a much smaller impact on the network performance in terms of packet size.

There is shown an upper limit on the block size, 2048 bytes in our case. This limit is determined by the underlying protocol structure. The X protocol does not put a constraint on itself. It relies on the underlying network and assumes a reliable network connection is available. Once the connection is established, it sends as fast as possible data units without waiting for acknowledgements. Before being sent over the network each X data block is divided into several packets. The fragmentation is done by the TCP/IP protocol based on the lower–level network structures, protocol headers, etc.. The Ethernet data format allows only 1500 bytes per packet, certainly not enough for 2048 bytes. At the destination, an assembly procedure takes place to restore the X data blocks. The fragmentation adds some overhead to the data at both the TCP/IP and the Ethernet layer as shown below:

Ethernet overhead	TCP/IP overhead	X protocol data unit	Ethernet overhead

On the other hand, the X window system is intended to be used in a general Internet environment. The client and server may communicate via several different networks. Also the IP datagram may take different routes to the destination point. The TCP protocol at the sending point does not have the knowledge of the physical network properties. Therefore it is hard for TCP to decide the best maximum packet size. The specification suggests a maximum size of 536 bytes for TCP packets, if it is used in a heterogeneous network environment. The Ethernet packet statistics are calculated in Table A4.4, where the bytes/pkt is the pure X protocol data per packet, excluding the overhead of TCP/IP and Ethernet. The standard overheads for TCP/IP and Ethernet are assumed, i.e., 40 bytes for TCP/IP and 26 bytes for Ethernet.

Table A4.4 Ethernet packet statistics (including both the client and server data)

applications	*xterm*	*xterm2*	*plaid*	*ileaf5*
mean pkts/block	4.38	2.55	12.06	3.03
mean bytes/pkt	211.98	15.26	162.82	124.45
mean pkt length	277.97	81.26	228.82	190.45
overhead %	24	81	29	35

The packet lengths are typically around a few hundred bytes, even under a heavy load (*plaid* case). It is worth noting that the packets transmitted for the *plaid* application are almost all *requests* to the server. This is to say, for large amounts of *request* traffic over the network the overhead is still high. In other words, the amount of overhead is independent of the application. This behavior may be explained by the fact that shorter network delay requires shorter packet size, which in turn results in higher overhead.

X traffic and Ethernet traffic

The generated X traffic has two patterns roughly. One is burst mode, like most interactive data traffic. The server tracks the user's random inputs with KeyPress, KeyRelease, etc., then sends events to the related client to trigger the transmission of drawing information of variable length. The other mode is when the client continuously executes a computation with some sort of algorithm and sends the results to the server to display, such as with the *plaid* program. The data rate is relatively constant, or with a small variance from the mean. To accurately characterize the bursty traffic is a difficult job. It may be left for further study. Here, we use the observations as well as the mean values to get some representative figures.

The instantaneous traffic intensity is determined by the amount of data arriving and the instant interarrival time. A detailed observation of the data captured by *xscope* shows that one X data block was generated and sent by the client side about every 0.103 seconds when displaying a large text file (in *xterm* case), and about every 0.06 seconds when displaying plaid patterns. In both cases, the number of bytes in a data block was very close to the upper limit, 2048 bytes, and with very small variance. This implies that the transmission rate was determined by the client program, not the protocol itself, in the measurement. The corresponding data rates are 159.068 kbits/s(*xterm*) and 273.067 kbits/s(*plaid*), respectively. This is calculated by multiplying the mean bytes per X data block with the mean data block generation rate. The mean values of X traffic are calculated in Table A4.5. The arrival rates in the first two cases indicate the user's interactive speed in front of the terminal. To determine an accurate measurement, the procedure must be conducted over a long period of time. For the machine–generated traffic, the arrival rate reflects the speed of the software and the hardware associated with the client and server.

Table A4.5 mean values of X traffic (including both client and server)

applications	*xterm*	*xterm*2	*plaid*	*ileaf*5
mean block arrival rate (blk/s)	4.93	8.77	15.38	9.09
mean block size	928.47	38.91	1963.60	377.07
traffic intensity	36.62 kbits/s	2.73 kbits/s	238.28 kbits/s	27.42 kbits/s

The resultant network traffic is calculated by multiplying the mean packet size, derived from the application level, with the packet rate measured at the network level. The results are summarized in Table A4.6. The traffic differences from the network level to the application level indicate the overhead of the TCP/IP and Ethernet protocols, plus some statistical errors.

Table A4.6 Ethernet traffic

applications	*xterm*	*xterm*2	*plaid*	*ileaf*5
traffic intensity	46.94 kbits/s	14.03 kbits/s	342.22 kbits/s	39.22 kbits/s
network util. %	0.47	0.14	3.42	0.39

The heaviest network traffic is 342.22 kbits/sec., which corresponds to 3.42 percent of the 10Mbits/sec Ethernet capacity. Assuming the Ethernet can handle an overall load up to 20 percent of its capacity without degrading performance, 6 *plaid* applications, or 50 *xterm* applications can run simultaneously over the network. Beyond that, the window performance constraints may shift from the machine to the network. It should be noted that since each X application is a random process with respect to its traffic, the mean values may not be sufficient for its description, especially for a widely spread distribution. In that case, simulation may be required. Another issue is that the traffic applied to the network is also a function of the machine speed. Software and hardware upgrades would cause varying increments of traffic intensity to the network.

Brief summary

Operating in the TCP/IP and Ethernet environment, X window applications do not generate much traffic to the network. The X performance is not restricted by the network. The traffic overhead due to the TCP/IP and Ethernet protocols is quite high, even though the X protocol aggregates data units into larger blocks before transmission. These blocks are broken down into small pieces at the lower layers to help reduce network delays. As long as the traffic volume is kept small, the overhead is not a big deal. Graphical plots of the four applications are given in Figures A4.6 – A4.9.

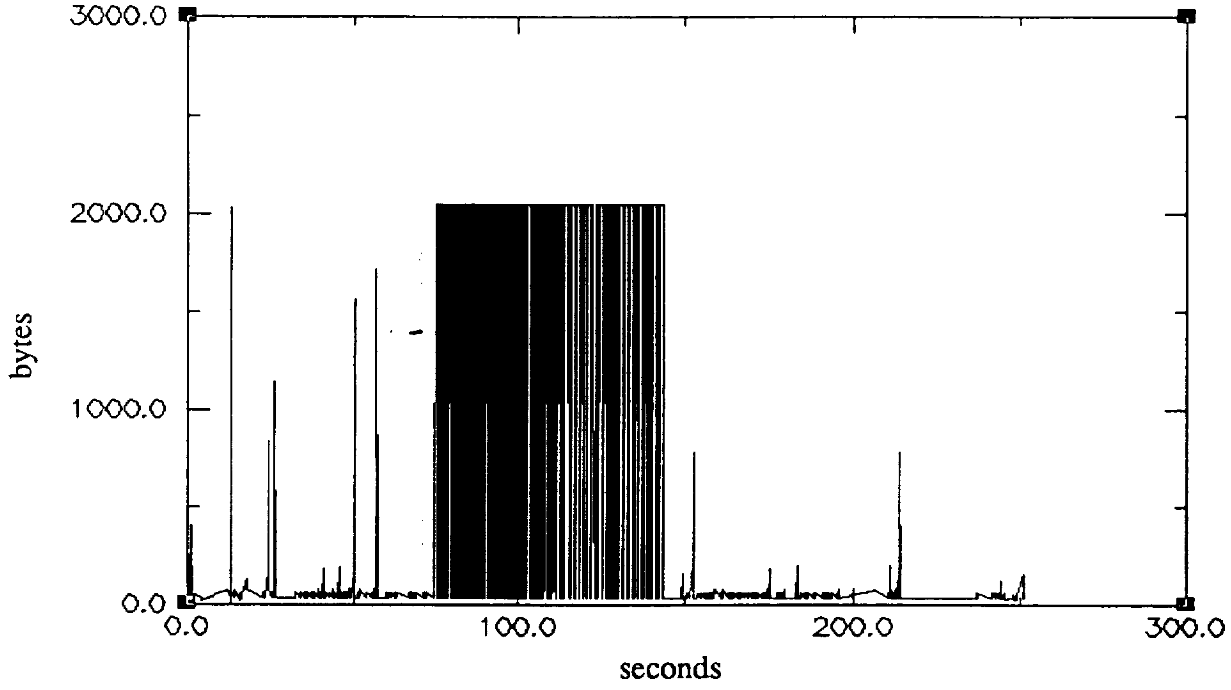


Figure A4.6.a Xterm

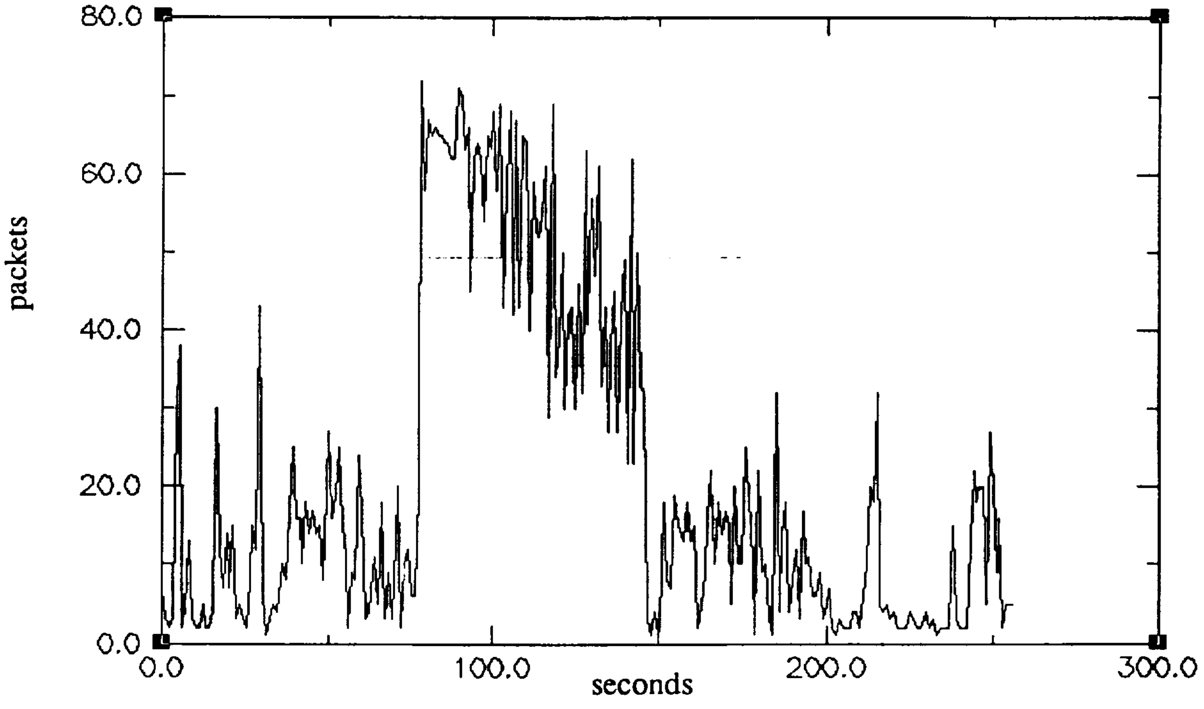


Figure A4.6.b Xterm

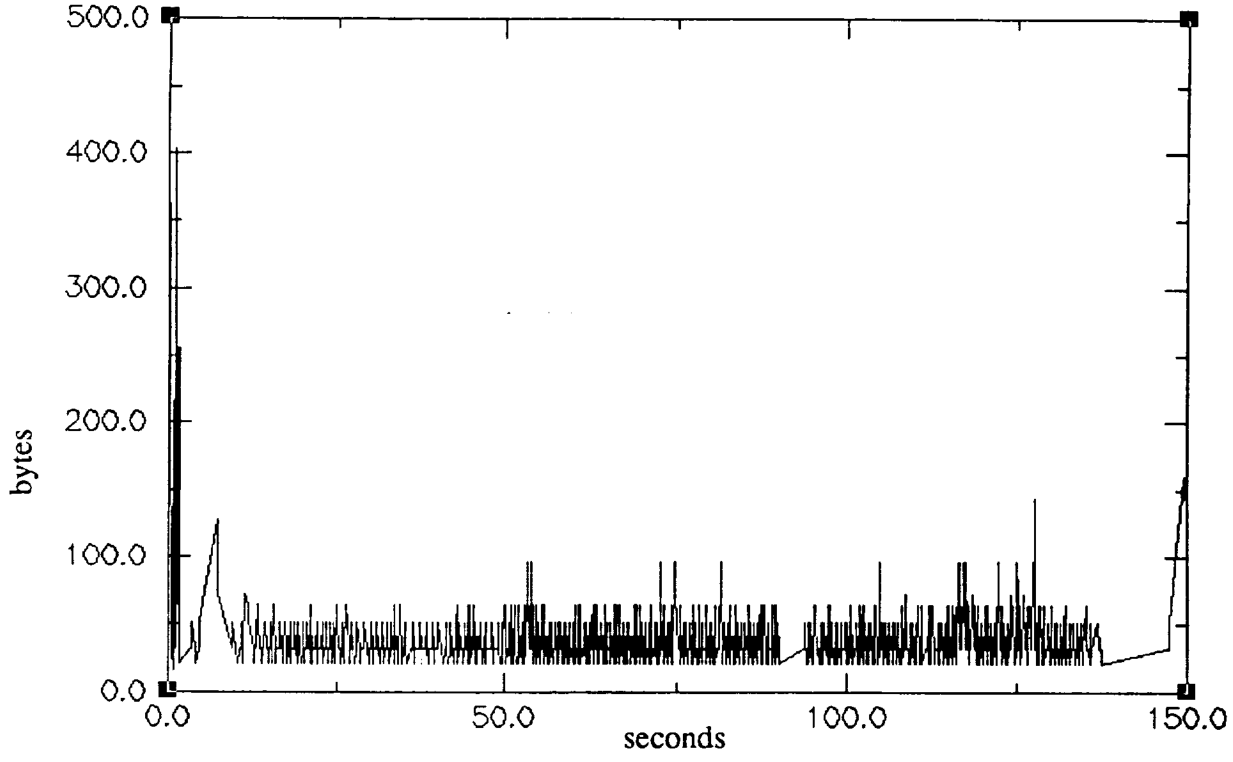


Figure A4.7.a Xterm2

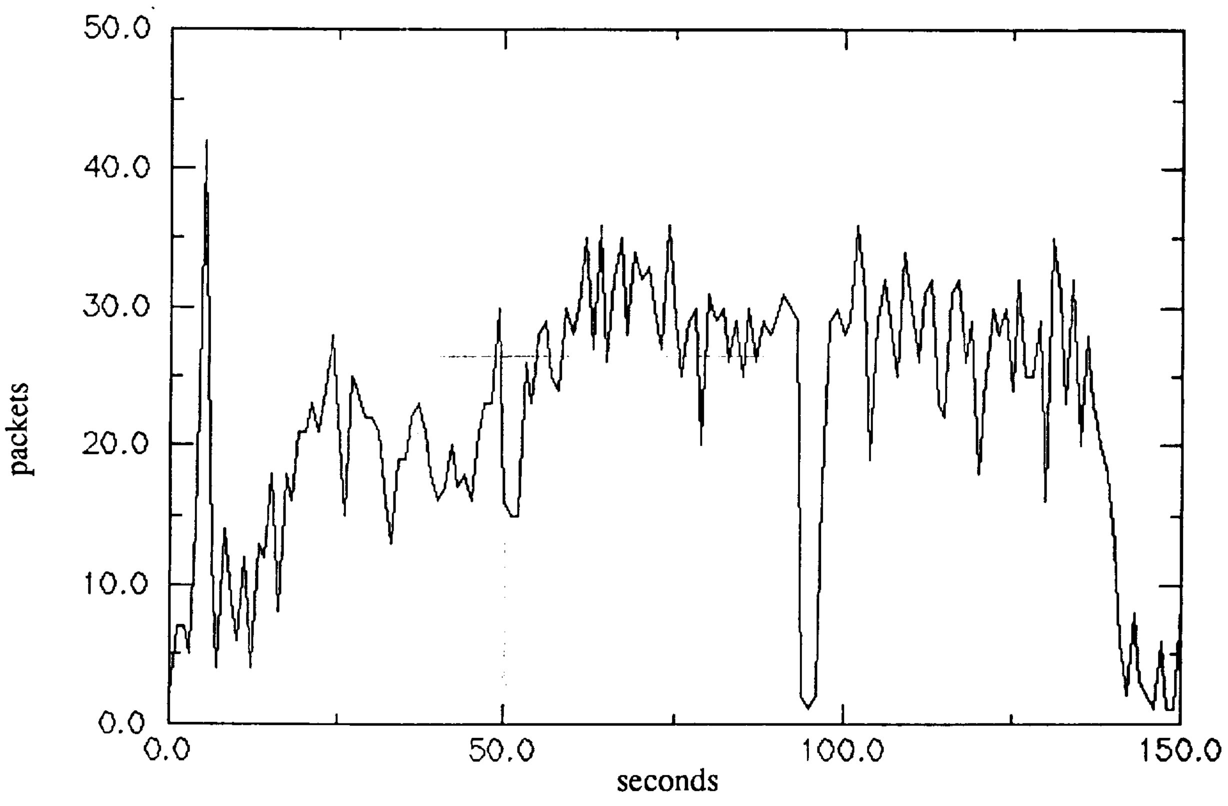


Figure A4.7.b Xterm2

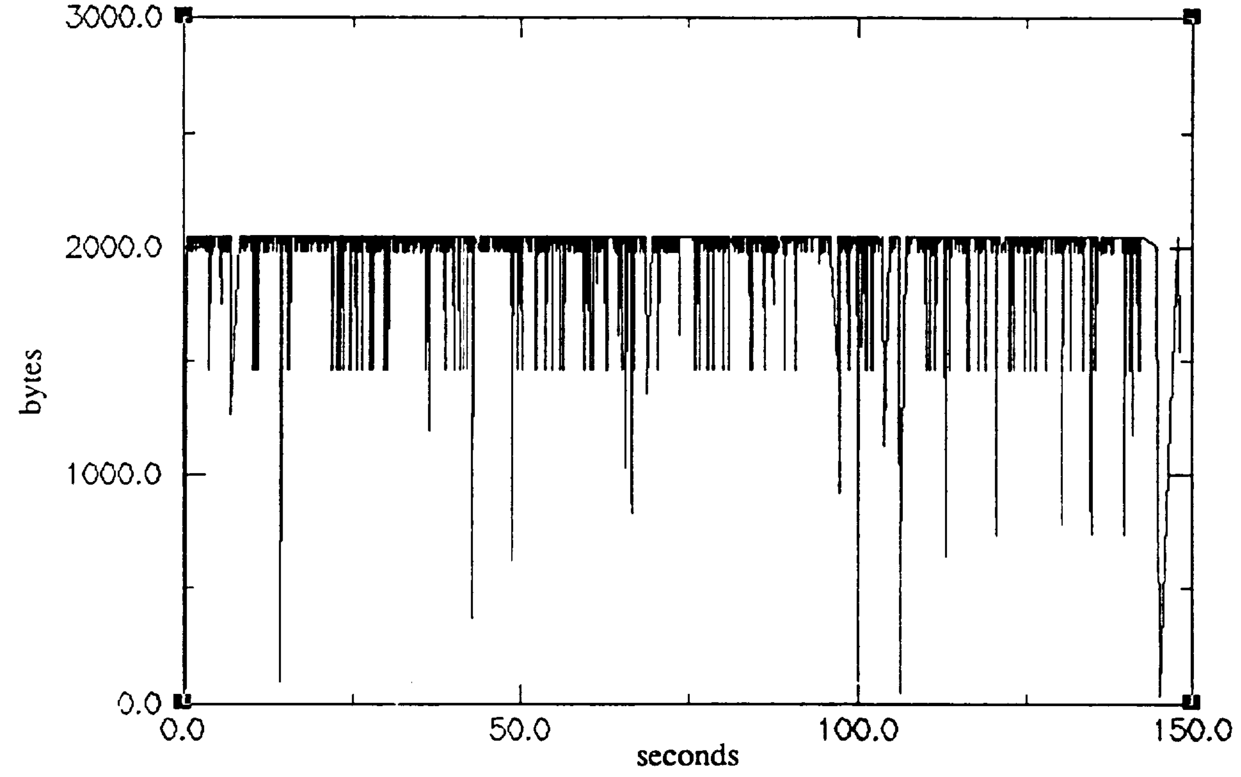


Figure A4.8.a Plaiu

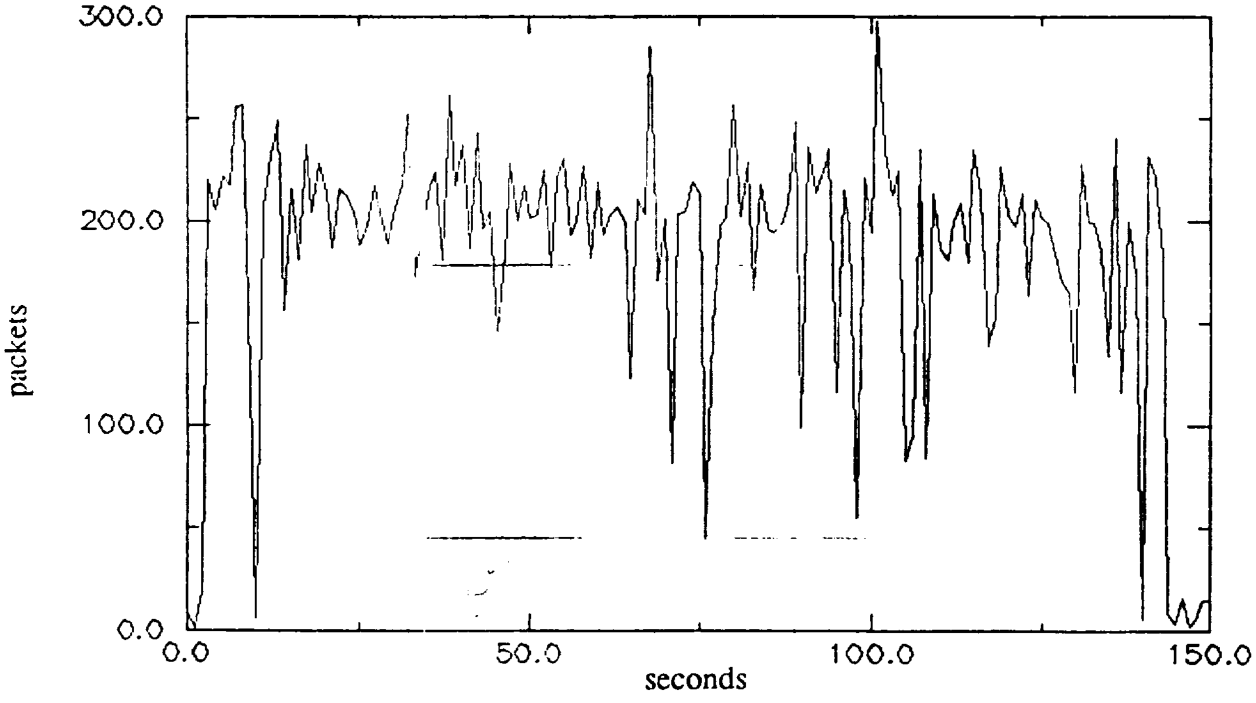


Figure A4.8.b Plaid

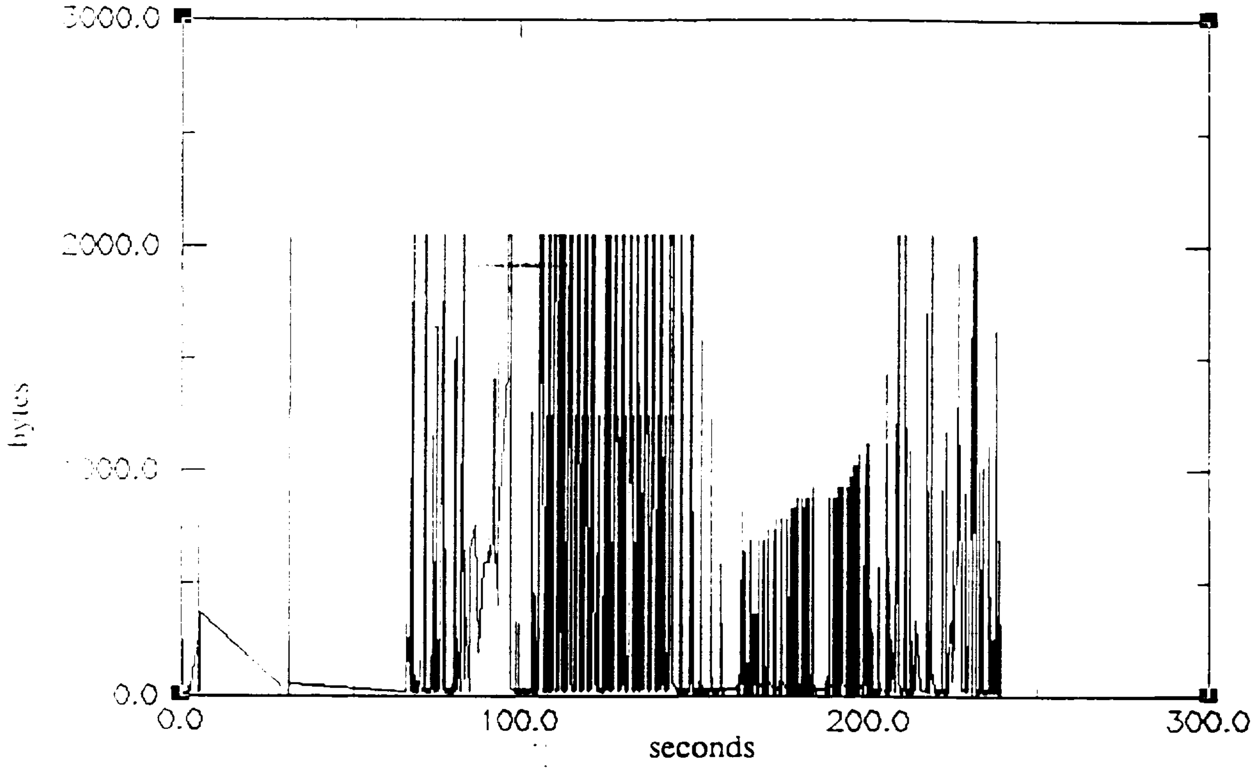


Figure A4.9.a Ileaf

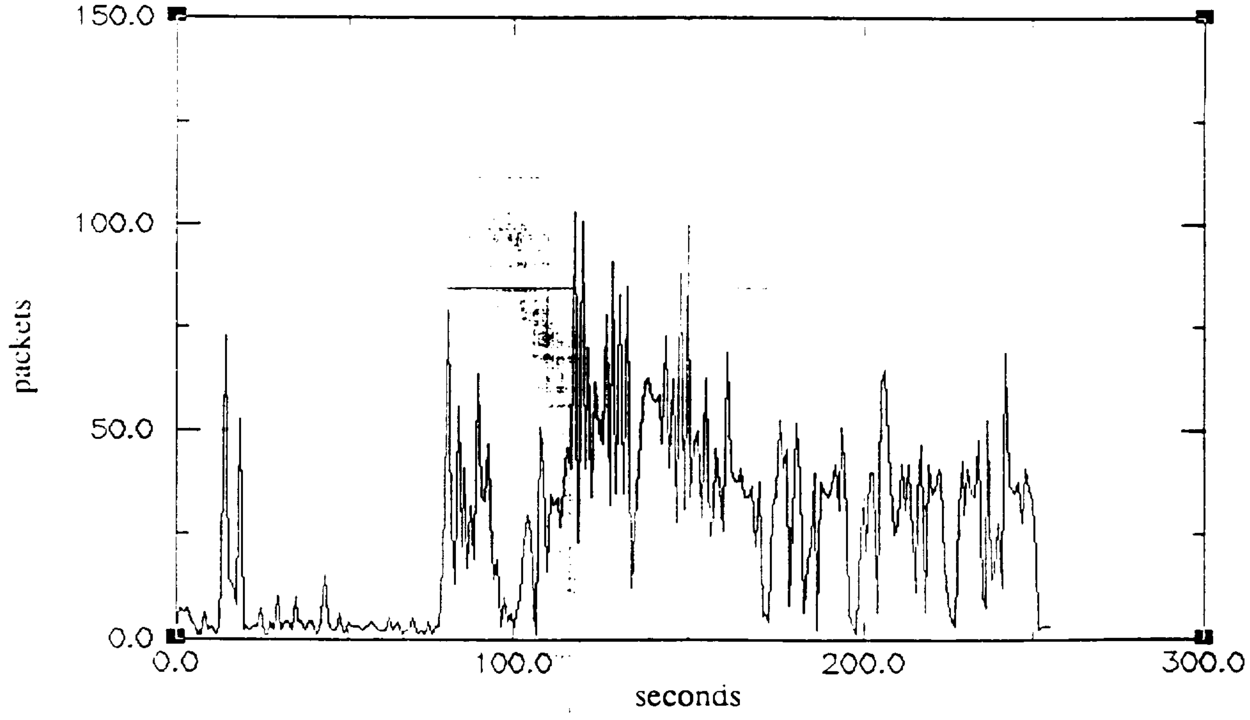


Figure A4.9.b Ileaf

APPENDIX
A5. X APPLICATIONS WITH MACINTOSH

Much interest has been shown recently in optimizing X applications to make use of much less expensive personal computers as the X platforms instead of powerful but relatively expensive workstations. The advantage is that it provides PC users with the access to networked supercomputers while still maintaining all the PC's application features so that it takes full advantages of network resources and the PC's merits. Compared with X terminals, PC's do not have to live totally at the mercy of the central systems and are free of the inconveniences caused by the central systems maintenance schedules. In addition, the PC market offers a variety of selections for different users needs, such as fast computation, low cost, or high–resolution display. This wide range of choices makes it easier to achieve a good performance/cost ratio.

In this section, we present a survey study on Apple's Macintosh machines and different networking components with emphasize on performance aspects of X applications over Ethernet.

Network Configuration

The X applications with Macs include running Mac X servers on the Macintosh machines, and using X's cross–network capabilities to connect to remote clients, so that the Mac user can run applications on remotely connected workstations or minicomputers. Another possibility is the reverse situation: X applications (clients) are running on the Macs and results are displayed on remote workstations through the network. The first application is generally of greater interest.

Figure A5.1 presents a general networking situation when running X windows on networked Macs. The components involved include the X servers, network protocols, interfaces between the Macs and the network, and the network itself. A router/bridge may be used if the X client/server connection is being made to another network with different networking technology. Assuming that the machines running the X clients or other specific programs — with results being displayed on local Macs through the X window system — are much more powerful (such as DEC's VAXcluster, Sun or HP

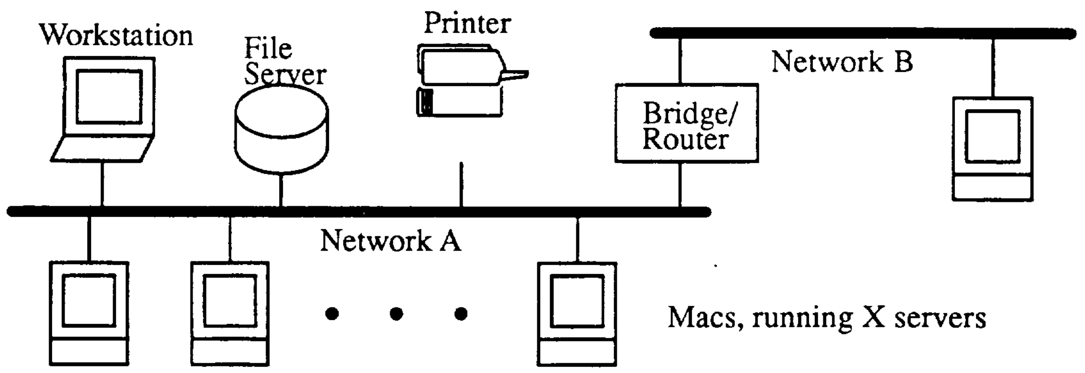


Figure A5.1 X Windows on Networked Macs

workstations), primary performance considerations of X applications should be given to the software and hardware elements in the Macs and in the network.

X server

There are two main X servers currently available for Macs: Apple's MacX and White Pines Software's eXodus. MacX complies with the MIT X Window System specification X11R4. It includes a built–in window manager that complies with the ICCCM standard. The window manager provides title bars, close boxes, and other Mac window controls. MacX also offers support for other window managers, such as OSF/Motif's *mwm*, or *twm*.

The latest version of MacX is MacX 1.1.7.

System requirements to use MacX by Apple include:

– Any Mac computer

– At least 2 megabytes memory

– At least two floppy disk drives (for Mac operating system 6.0.5 and later)

– A hard disk (for Mac operating system version 7.0)

– Mac operating system version 7.0, or version 6.0.5 and later

– A network connection (LocalTalk or an Ethernet connection)

Apple's UNIX system for Mac, A/UX 2.0, is an alternative operating system. In fact, A/UX has the MacX incorporated in it, so that it provides a native X environment.

To use Ethernet connection, it needs Apple's Ethernet NB card or equivalent products from other vendors. The transport protocols supported by MacX include MacTCP (Apple's implementation of TCP/IP protocols), AppleTalk ADSP (AppleTalk Data Stream Protocol), and DECnet. The shipment of MacX includes the MacTCP package.

The performance of MacX is affected by its working environment, such as the speed of the Mac the MacX runs on and the operating system used. A test of performance efficiency for MacX and eXodus was conducted in [21] by using a standard X window benchmark program that ran all the integral X graphics algorithms in turn and then produced the statistical benchmark measured in Xstones. The Xstones represent the number of graphic operations per second. The MacX was running on a Mac IIci under the Mac operating system. The results are compared with those of MacX running under A/UX, and a Sun SPARCstation 2 using its own server:

	MacX under Mac OP	MacX under A/UX	Sun SPARCstation 2
Xstones	8285	13796	19374

Network protocols

AppleTalk is the network architecture Apple developed for Macs to talk to each other and to share printers and servers. AppleTalk is a set of protocols at different OSI layers. Originally AppleTalk only contained one data link protocol, called LocalTalk Link Access Protocol, which works on top of Apple's LocalTalk cabling, normally twisted–pair cables or phone wires. Each Mac product has the built–in functionality of LocalTalk. Thus it is easy for Macs to form a LAN over the existing phone wires in a building. The major drawback of LocalTalk is that its speed is rather slow: 230,400 bits per second. Some third–party products can be used to speed up the LocalTalk transmission. For example, Dayna Communication's DaynaTalk and TOPS's FlashTalk offer some performance improvements to running AppleTalk over LocalTalk cabling. The tests in [23] indicated an improvement of anywhere from 25 percent to 50 percent in file–transfer times. Even so, FlashTalk's top speed of 768 kbits per second and DaynaTalk's 850 kbits per second are still far below the Ethernet capability.

In 1989, Apple introduced AppleTalk Phase 2 to cope with the growing network [22]. AppleTalk Phase 2 did nothing for LocalTalk, but added Ethernet support and Token Ring support in the AppleTalk protocol suites. Also an extended addressing scheme was included to remove the original AppleTalk's 254–device limitation on the Mac network. To move to higher–speed networks, Ethernet has become an inevitable alternative to the slow LocalTalk. Ethernet provides the ways for users to quickly transfer intensive volume of data, such as large graphics files, so it is the natural solution for the bit–mapped X window applications. In addition, Ethernet is the most popular medium for networking UNIX workstations, IBM PCs, and DEC's VAX computers. Hooking Macs on the Ethernet makes it easier for Macs to access those other computers.

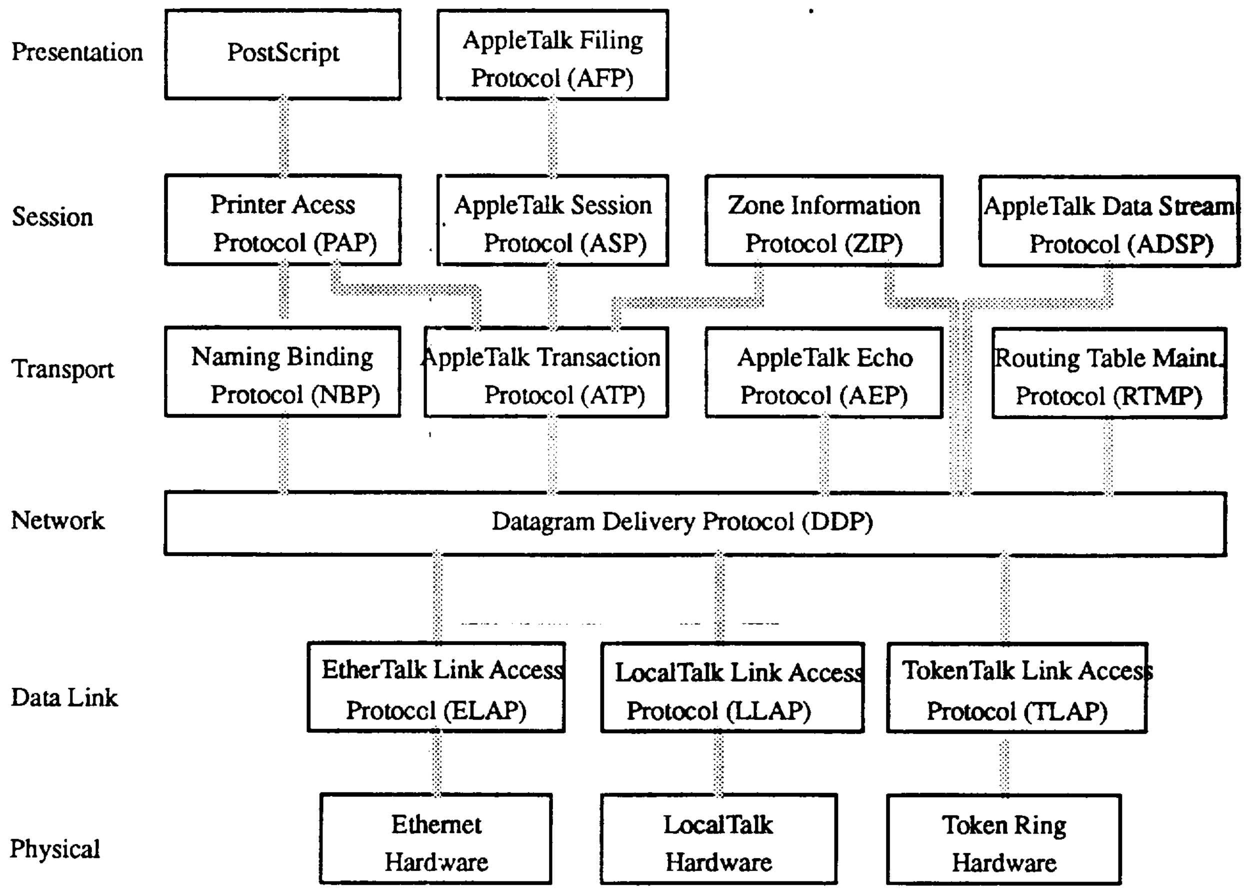

Presentation
PostScript
AppleTalk Filing Protocol (AFP)
Session
Printer Acess Protocol (PAP)
AppleTalk Session Protocol (ASP)
Zone Information Protocol (ZIP)
AppleTalk Data Stream Protocol (ADSP)
Transport
Naming Binding Protocol (NBP)
AppleTalk Transaction Protocol (ATP)
AppleTalk Echo Protocol (AEP)
Routing Table Maint. Protocol (RTMP)
Network
Datagram Delivery Protocol (DDP)
Data Link
EtherTalk Link Access Protocol (ELAP)
LocalTalk Link Access Protocol (LLAP)
TokenTalk Link Access Protocol (TLAP)
Physical
Ethernet Hardware
LocalTalk Hardware
Token Ring Hardware

Figure A5.2 Appletalk Protocol Suites

The transport protocols used in Macs to support MacX are AppleTalk ADSP, MacTCP, or DECnet. DECnet transport protocol is, of course, from another companies. MacTCP is shipped with each purchase of MacX. Therefore, MacTCP is the default transport protocol for MacX. It allows the Macs to connect to any existing TCP/IP network. To use MacTCP, each Mac needs an IP address from the network administration. MacTCP can run under AppleTalk with Ethernet connection.

Ethernet interfaces

In the past years, the number of Mac Ethernet interfaces has proliferated, and costs have come down. These Ethernet interfaces fall into two categories: cards that plug into a slot inside the Mac, and external adapters that plug into the Mac's SCSI port. The internal interface cards are connected in the Mac either to the NuBus slot or PDS (processor direct slot), depending on the Mac's type. SCSI stands for *Small Computer Systems Interface.* SCSI enables the Mac to communicate with high–speed peripherals, such as hard drives, image scanners, personal laser printers, CD ROM drives, and tape backup devices. Externally, SCSI is a port designed to attach additional hardware to the Mac. It works as a parallel bus, transmitting 8 bits at a time. A new version of the SCSI, SCSI–2, enhances the bus transmission performance by expanding SCSI's parallel data path from 8 bits to 16 bits, making a possible data transfer rates of 10–40 megabytes per second. The devices connected to SCSI are generally in a series configuration, called *daisy–chaining*. That means the SCSI bandwidth must be shared among these devices. Therefore, Mac's SCSI bus is generally slower than both the NuBus and PDS. For example, the 32–bit NuBus has a maximum quoted performance of 37.5 megabytes per second. However, the SCSI connection is compatible to all types of Macs provided that they have a SCSI port available.

There are four types of cables for Ethernet: 10BASE–5, 10BASE–2, 10BASE–T, and fiber–optic. 10BASE–5 is a thick coaxial cable with the segment length of no longer than 500 meters. It is best used as a network backbone. To attach a Mac to the thick coax usually requires a transceiver from the Ethernet interface card. 10BASE–2 stands for a thin coaxial cable with a segment length of up to 200 meters. The thin coax is less expensive and more flexible. Most Mac Ethernet interfaces have built–in transceivers for thin coax, so there is no need for external transceivers. But Apple's Ethernet LC card and Ethernet NB card have no transceivers on board. They include a special socket, called Apple Attachment Unit Interface (AAUI). So, using these two Ethernet cards does require an external transceiver. One of the advantages of using AAUI is that one does not need to change the Ethernet card when switching Ethernet cables, say, from 10BASE–2 to 10BASE–T. 10BASE–T runs over twisted–pair phone wire and usually uses a star topology. A 10BASE–T cable length (the distance allowed between a node and the star hub) is limited to 100 meters. The star topology makes 10BASE–T wiring easier to modify than its coaxial cousins, because the star hub (concentrator) resides at a central location. Each Mac must have either an Ethernet interface with a 10BASE–T transceiver built in or a built–in AUI (Attachment Unit Interface) port connected to a separate external 10BASE–T transceiver. The fiber–optic cable is the choice for the future. It outperforms the coax in many ways. But currently only a few Mac Ethernet interfaces support it, e.g., Mac2000 Board from Network Resources.

A list of main Ethernet connection interfaces is tabulated in the following tables [24, 25, 26, 27].

Boards

Interface	Vendor	Mac	thick	thin	tw. pair
EtherLink/NB	3Com	IIs	x	x	x
EtherLink/NB TPX	3Com		x	x	x
Ethernet LC Card	Apple	LC	x	x	x
Ethernet NB Card	Apple	IIs	x	x	x
MacCon+	Asanté Technologies	IIsi, SE/30	x	x	x
FriendlyNet Card	ditto	LC		x	x
LanWay Ethernet	Avatar	LC	x	x	x
E4010–E6020	Cabletron Systems	IIs, SE/30	x	x	x
GatorCards/E	Cayman System	IIs	x	x	x
Ether2/Ether DS	Compatible Sys. Corp	IIs	x	x	x
DaynaPort E	Dayna Comm.	IIs, LC, SE/30, SE	x	x	x
FastNet	Dove Computer Corp.	IIs, SE/30, and SE	x	x	x
PhoneNet Card for Ethernet	Farallon Computing	IIs, SE/30	x		x
Magic Ethernet II	MacProducts USA	IIs	x	x	x
EtherNode 16	National Cemiconductor Corp.	IIs, SE	x	x	x
EtherNode 32	ditto	SE/30, IIsi	x	x	x
Mac1000	Network Resource	IIs, SE	x	x	x

NuvoLink II	Nuvotech	IIs	x	x	x
MacConnect	Racal–Datacom	SE, SE/30, IIsi	x	x	x
EtherPort	Shiva Corporation	IIs, SE, SE/30	x	x	x

Note: IIs means IIsi, II and up.

SCSI type

interface	vendor	mac	thick	thin	tw. pair
Asanté EN/SC	Asanté Technologies	all	x	x	x
Ether+	Compatible Sys. Corp	all	x	x	x
DaynaPort	Dayna Comm.	all	x	x	x
NuvoLink SC	Nuvotech	all	x	x	x

Over the years as Mac networks moved from the slow LocalTalk to Ethernet, many studies have been made on the performance of Ethernet interfaces to investigate the possible potential bottleneck from the Mac to the Ethernet highway. Here we present the most interesting results from two tests [26, 27].

In [26] 15 twisted–pair Ethernet boards for Macs were tested. These boards include: 3Com's EtherLink/NB, Apple's Ethernet NB Card, Asanté Technology's MacCon3 for NuBus, Avatar's LanWay Ethernet, Cabletron's E6000 DNI, Compatible Systems's Ether2, Dayna Communications's DaynaPort, Dove Computer's FastNet III N, Farallon's PhoneNet Card for Ethernet, National Semiconductor's EtherNode 16 NB, Network Resources Corporation's Mac1000, Racal–Datacom's MacConnect Ethernet Card, Shiva's EtherPort II, Sonic Systems's Sonic Ethernet Series, Technology Works's Ethernet Kit. In the tests, two Mac IIci's were connected over Ethernet, and executed file transfers between themselves. The transfer rate was timed for each Ethernet board. To eliminate the limiting factor of the slower hard drive, a RAM disk was used instead to store the files. Thus the affecting elements are the Ethernet interface and the processor of the Mac. The results showed very little difference in the transfer rates among the tested boards when transferring a 5MB file. The average rate is about 1.5 Mbits per second. Smaller file needed relatively longer time to transfer, due to the fixed overhead of the control packets. A comparison between the RAM disk and hard drive was also tested. Not surprisingly, RAM disk provided much better Ethernet performance, which indicates that the hard drive, if it is involved in the connection, is probably the bottleneck. Although the 1.5 Mbits per second result does not separate the Ethernet board from the processor or prove that the Ethernet board might not be saturated, it does show a 1.5 MB rate is obtainable for the Ethernet boards.

Another test was carried in [27] about two years ago. It tested different Ethernet boards, and SCSI adapters on Mac IIs, SE/30, SE, and Plus. The results showed the performance of Mac IIs, and SE/30 with internal Ethernet boards (NuBus, PDS) is much better than SEs with Ethernet boards, or SE/30 with SCSI adapters. It reveals that the CPU or the SCSI port is the performance bottleneck, not the Ethernet interface. A further effort was made to test the interface raw performance. The raw performance is the fastest speed the interface can transfer data without being held up by the CPU or the SCSI. Network General's Sniffer together with a specially developed NetBasher! by MacUser NetWorkShop was used for this test. The maximum data transfer rates measured by the Sniffer were about 4.0–4.8 Mbits per second, where Asanté's MacCon+, Racal–Datacom's MacConnect, Cayman's GatorCard, and Cabletron's E6000 outperformed the others.

There are other tests on the Ethernet interfaces, some using client/server file transfer, some using special programs to generate data traffic to avoid accessing external devices. None of those tests observed the bottleneck at the Ethernet interfaces.

Mac machines

For an X server running on a Mac, choosing an appropriate Mac also means having better performance. The Macintosh family consists of many combinations at different prices to cater for different demands. The processor speed, memory size, bus configuration, etc. all contribute to the overall performance of a Mac. Most of the Macs in the current market include either M68030 or M68020 microprocessors, with a clock speed of 16 MHz to 32 MHz. The M68020 has an instruction cache, while the M68030 has both an instruction and a data cache. A cache is an area of memory that can be directly accessed by the CPU without the delays involved in reading from main memory, thereby increasing the effective processor speed. The profiles of various Macs are summarized in the Appendix.

Mac LC is the least expensive color Macintosh. It uses a 68020 microprocessor running at 16 MHz. It does not include a math coprocessor, the trade–off for the low price. But it has a built–in color video circuitry capable of generating 256 colors. To take advantage of the 256 colors, Apple introduced a sharp, low–cost 12–inch color RGB monitor, with 512x384 pixels. The LC has a standard 2 MB RAM expandable to 10 MB, to accommodate for the Mac operating system version 7.0. But the memory–management unit is not included. Without the MMU, the LC cannot utilize the virtual memory feature provided in Mac operating system version 7.0.

The LC II looks very much like the LC on its exterior. And it sells for nearly the same as or less than the LC. The big difference is that a 68030 microprocessor, instead of 68020, is included. There is not much of performance difference between LC and LC II. But the LC II offers virtual memory capability under the Mac operating system 7.0.

Quite a few tests of Ethernet interfaces used Mac II series computers, of which Mac IIsi is considered to have the better performance/cost ratio. The IIsi is faster than the LC. It can accept a NuBus expansion board, of which there are considerably more than there are LC boards. And the IIsi moves data between the memory and the processor in 32–bit portions instead of 16–bit as the LC II does. The IIsi can run Apple's A/UX version of the UNIX operating system. But IIsi costs hundreds of dollars more.

The fastest of Mac the II series is the IIfx. Its M68030 microprocessor runs at 40 MHz. However, is the most expensive.

Other factors

The operating system is another factor that needs to be considered for X window applications. Besides the Mac operating systems, Apple has a UNIX operating system, A/UX 2.0, for Macs. A/UX has the built–in MacX package, and tests in [21] indicate that running MacX under A/UX provides better performance. A/UX supports standard TCP/IP communication over Ethernet to other UNIX machines. To use TCP/IP protocols, there needs to be an Ethernet card installed in the Mac and configured to its TCP/IP mode. A/UX also provides AppleTalk connectivity over LocalTalk or EtherTalk networks. If both the LocalTalk and Ethernet cards are installed, you can choose between these networks. But A/UX requires a faster Mac, and at least 8 MB memory in order to achieve this better performance. If the Mac users do not need many UNIX features and networking services, the investment for A/UX would not be economically justified.

If the X clients are going to run on a different network than the one to which the Macs are connected, a router may be used to connect these networks. In this case, the X client/server connection involves data passing through the router, which should be taken into account in the overall performance assessment.

Brief summary

Multiple factors must be considered for running X servers on Macs. Tests show that the Ethernet network has sufficient bandwidth to support X applications. The Ethernet card should not be the bottleneck for connecting Macs to Ethernet given to the raw data traffic it can support, which means the real world applications must be able to generate enough traffic to saturate the Ethernet and the Ethernet cards. Using faster Macs may help improve the X window performance. And more memory size is generally a good choice to reduce the use of slower hard drives. A/UX provides a better environment than Mac O.S., but costs more.

Mac Profiles

Mac II series

	Mac II	Mac IIx	Mac IIci	Mac IIsi	Mac IIcx	Mac IIfx
Processor	68020	68030	68030	68030	68030	68030
Math co-processor	68881	68882	68882	68882 (option)	68882	68882
Clock speed	16MHz	16MHZ	25 MHz	20MHz	16MHz	40 MHz
Max. RAM	8 MB	8 MB	32 MB	17 MB	8 MB	32 MB
Process cache	instruction	instruction and data	instruction and data, 32K cache	instruction and data	instruction and data	instruction and data, 32K cache
ROM	256K socketed chips	256K SIMM	512K SIMM	512K SIMM	256K replaceable	512K SIMM
Color/gray–scale support	yes	yes	yes	yes	yes	13–inch, 640x480 RGB, 24–bit color card
Internal expansion slot	6 NuBus	6 NuBus	3 NuBus	1 NuBus	3 NuBus	6 NuBus

Mac Plus, Mac SE, Mac SE/30, Mac Classic series, and Mac LC series

	Mac Plus	Mac SE	Mac SE/30	Mac Classic	Mac Classic II	Mac LC	Mac LC II
Processor	68000	68000	68030	68000	68030	68020	68030
Math co-processor	no	no	68882	no		no room	no room
Clock speed	7.83MHz	7.83MHZ	15.6672MHz	8 MHz	16 MHz	16 MHz	16 MHz
RAM	4 MB	4 MB	8 MB	4 MB	10 MB	2 – 10 MB	4 – 10 MB
Process cache	no	no	instruction and data	no	instruction and data	instruction	instruction and data
ROM	256K socketed	256K socketed	256K SIMM	512 K	512 K SIMM	512 K socketed	512 K socketed
Color/ gray–scale support	no	no	yes	no	yes	yes, 16–bit color	yes, 8–bit, or 16–bit color

APPENDIX

A6. DECnet Capabilities

Introduction to DECnet

DECnet is a proprietary network facility developed by Digital Equipment Corporation. It has undergone five phases in development from phase I through the current phase V. Most of the earlier phases were developed around proprietary DEC protocols and services. Phase V has seen a migration to providing more Open Systems Interconnection (OSI) services and protocols in addition to an upgrading of the phase IV services. Since phase V was designed to remain compatible with earlier phase IV, a short look at phase IV makes it easier to understand the components of phase V. Phase V is also sometimes called Advantage–Networks.

Phase IV DECnet is generally centered on work groups that are connected together with Ethernet facilities. These work groups are then interconnected with wide area bridges or routers. Within the work groups, the standard Ethernet protocols are utilized. For wider distribution, Digital's Digital Data Communications Message Protocol (DDCMP) is usually used. Above these data link protocols are the DEC proprietary network, transport and session layers. On top of the session layer are several applications, two of the most common of which are the Data Access Protocol (DAP) and the Command Terminal (CTERM) protocol. DAP provides access to remote data bases, and CTERM is a remote terminal program, very similar in nature to the Telnet feature of TCP/IP.

These two services are supplemented by a number of other services, including a messaging system over DEC's Mailbus. Other services are provided for videotext, remote consoles, booting diskless nodes, bulletin boards, etc.

Phase V continues to include all the items from phase IV, including Ethernet, the IEEE 802.3 version of Ethernet, DDCMP, the permanent virtual circuit portion of X.25 networks, and the IEEE token bus protocol. In addition, phase V has added services supporting the ISO High–Level Data Link Control (HDLC) protocol, as well as the Fiber Distributed Data Interface (FDDI) protocol. Phase V also introduced two new proprietary DEC protocols that allow public data networks to be incorporated into DEC environments. These services are the Modem Control protocol (physical layer), and the Dynamically Established Data Link (network layer). These services permit dynamic establishment of modem connection through a public switched network using X.25 packet switched networks. For LAN's, DECnet phase V supports the Logical Link Control (LLC) protocols that may use either Ethernet, FDDI or token ring.

At the network layer, DEC uses ISO standard format 8473 for data and error packets. Hence interoperability with non–DEC equipment is maintained through this standard. Network routing is determined by a combination of the ISO 9542 standard and some additional DEC proprietary routing software. The 9542 protocol is primarily concerned with routing between End Systems and Intermediate Systems (ES–IS), while the DEC routing algorithms deal with Intermediate System – Intermediate System (IS–IS) routing. Non–DEC equipment that is not ISO 8473 compliant can be connected to a DECnet by use of a common subnet such as X.25.

At the transport layer, DEC uses the Network Services Protocol (NSP). This protocol is similar to TCP or the ISO TP4 protocols. Phase V also supports two other ISO protocols, TP0 and TP2. TP4 is used primarily for DEC applications, and the other two for non–DEC systems.

Up through the network layer, DECnet phase V is a fairly conventional OSI system with the inclusion of DDCMP to maintain compatibility with phase IV, and MCS for extra features. At the Session layer, there is a distinct split, and DECnet provides two separate session layers. The OSI session layer is provided to support OSI applications. A separate session layer, Digital Session Control protocol (DSC) is provided to support DEC services and to provide interconnection to proprietary DEC services at the lower levels of phase IV and phase V.

With all these different options at many levels, it is obvious that there is an almost bewildering combination of protocol stacks that are available to the users. Almost any of these combinations could be used in certain instances. For each application, DEC keeps track of the protocol stack in use through something called a tower. A tower is a set of addresses from the network layer on up. Each node keeps a list of towers, representing the possible combinations of protocols that might be used for communication. When communications is desired between two nodes or applications, the session layer examines the tower sets, and comes up with a common set that can be used for communication. A part of the DSC called the Domain Name Service (DNS) provides a distributed naming service that permits applications to communicate across the network using a logical name.

The towers are used only within the DECnet domain. For other connections, the OSI session layer is used. On top of this layer, DEC uses File Transfer Access and Management (FTAM) protocol and X.400. The FTAM implementation also has something called an FTAM/DAP gateway which allows DEC networks to access non–DEC FTAM systems. DEC has also implemented gateways for Telex and the TCP/IP SMTP.

Ethernet and DECnet

The major user of the data link layer in a DECnet system is the Digital Network Architecture (DNA) network layer. In addition, there are two other users that do not necessarily access the data link layer through the network layer. These users are the Local Area Transport (LAT) and the Maintenance Operations Protocol (MOP). LAT is a direct user of the data link layer that is used to provide terminal to host transfer. MOP is used to download operating systems to a remote host for booting diskless workstations and PC's. In addition, other network services may share the data link with these protocols. This can include services such as the naming service, the time protocol, remote procedure calls (RPC) and TCP/IP.

Digital originally used DDCMP as the major protocol on their networks. This protocol is still used for some wide area communications, but it has largely been replaced with Ethernet for most local area communications environments. Most Digital equipment is delivered with Ethernet interfaces, and this includes not only computers, but printers and disk servers as well. A very large percentage of the communications within a DECnet takes place over an Ethernet physical layer. This is particularly true for local traffic.

An appropriate model of a DECnet phase IV system would include a number of devices interconnected locally with Ethernet facilities. A set of these groups of workstations would then be interconnected by DDCMP facilities. For a phase V system, this model would be changed to add a number of different alternatives for the WAN interconnections. These would include FDDI, X.25 and HDLC. The move from DDCMP to HDLC places DEC into position to interface with the soon to be very important Integrated Services Digital Network (ISDN). Phase IV also supported HDLC, but it has become much more important in phase V.

In some cases, it is desirable to connect several devices to the Ethernet at a single point. When this is the case, a multiport transceiver that operates more or less as a data concentrator can be used. The Digital equivalent of a multiport transceiver is called a Digital Local Network Interconnection or DELNI. This device is sometimes referred to as "Ethernet in a can."

MOP

Within DECnet, there are several support protocols that are of interest in the development of the Yellow Creek network facility. Some of these services are significant from the network standpoint, but are of no special interest in this paper. These include the DNA naming service, the time service and remote procedure calls service (RPC). One support protocol of particular interest at this point is the Maintenance Operations Protocol or MOP.

As was noted earlier, at the session layer and above, there are two distinct architectures in use in a Digital network. These are DECnet and OSI. To these, a third must be added, and this is MOP. Technically, MOP is part of the DECnet protocol, but from a practical standpoint, it is a separate architecture. MOP is a direct user of the data link layer, and can operate without any of the DECnet protocols. Typically, MOP uses an Ethernet link, but it can operate over HDLC or DDCMP.

MOP is a fairly primitive protocol, and contains very limited security features. While MOP offers three levels of functionality, including testing the communications link and operating as a remote console, the main use of MOP is in download operating systems to a diskless workstation. While this can be initiated from a console, it more commonly takes place when the workstation is initialized.

When a diskless workstation is powered up, it utilizes the MOP to broadcast an appeal for help. A node on the network will then volunteer help. The diskless workstation responds with a memory load message, and data messages transfer the required data from the volunteer station into the memory of the workstation. The diskless workstation starts with a primitive primary loader that is used to load a secondary loader into the workstation. The secondary loader then loads a tertiary loader that can load the operating system into the workstation.

Instead of loading the normal operating system, the tertiary loader can also be used to load the network management initialization script (CMIP) into the workstation. This script is used to set network parameters such as maximum data packet sizes and retransmission timers, etc. Once the diskless workstation has been booted, it then performs its communications tasks over the network like any other node on the network.

LAT

An important concept in the Digital environment is that of the VAX cluster. The purpose of the VAX cluster is to take a disk drive (actually two) and make it available to multiple users simultaneously. This is accomplished through internal communications protocols that are transparent to the network users.

A VAX cluster typically consists of two or more VAX minicomputers. These computers communicate via a DEC 70 Mbps dedicated system backbone called the CI bus. The computers also communicate over the same bus with two Hierarchical Storage Controllers (HSC). The HSC is a specialized disk server that is connected to two disk drives. The disk drives are dual ported to multiple controllers.

All the storage facilities in the cluster are duplicated to provide redundancy and a measure of fault tolerance. Data written to the disk is shadow written to both disks simultaneously. This also improves fault tolerance. A lock manager running on the VAX system coordinates access to the disks from all users.

The combination of the minicomputers, the CI bus, the HSC's, the disks and the lock manager constitute the VAX cluster. To the users of the system, all the different VAX systems look the same. They all have access to the same data and the same peripheral facilities. Therefore, most users don't care which system they log into, except that they would like to log in on the computer that is likely to give them the best service.

The DEC Local Area Transport (LAT) was originally developed by Digital to solve the problem of which system VAX system to access. Like MOP, LAT is a relatively low level protocol that accesses the data link layer directly without reliance on many of the architectural features of DECnet. Since LAT provides the capability to connect many different clients to many different servers, it has a much broader application that first envisioned, and can be used in a number of different ways.

LAT attempts to balance the load on the VAX minicomputers through the use of a rating system. When a new session is established, LAT attempts to select the node that has the best rating at that time, and will, at least theoretically, provide the best service. Unfortunately, if service deteriorates, the load is not re–balanced.

LAT also provides the capability for multiplexing messages from several users (through the terminal server) into a single message. Because of this multiplexing capability, LAT makes an excellent platform for an X windows system. The terminal server can combine the many asynchronous events associated with an X session into a single packet for transmission to the computer. This multiplexing avoids a lot of the interrupts associated with an interactive system, and greatly reduces the load on the CPU.

Since LAT accesses the data link layer directly, it assumes an underlying IEEE 802.2 (data link) and a multicast capability. LAT can operate over either a CSMA/CD network or FDDI, although Ethernet is more common. LAT is intended to utilize only a small percentage of the bandwidth on the network.

LAT is a master/slave system based primarily on the use of timers. This philosophy greatly reduces the complexity of the slave component. Because LAT doesn't utilize the upper layers of the DECnet architecture, it is a relatively simple protocol, and provides little in the way of security, fault tolerance or transport service. LAT depends on the underlying data link layer to provide some of these facilities.

There are two major layers in LAT: the virtual circuit layer and the session layer. The virtual circuit layer is concerned with the establishment, maintenance and disconnection of the virtual circuits between the server and the host, and provides those services normally associated with a virtual circuit. The session layer provides the user interface to the system. The session layer performs the multiplexing of data from several terminals into a packet for transport over the network. LAT also provides a directory service that manages names for ports, nodes and services on the system.

In the LAT system, the terminal server normally assumes the role of the master, and the host is the slave. Whenever a timer in the master expires, the master transmits data to the slave. The slave is then able to transmit data back to the master by piggybacking this data onto the acknowledge packet. In the event that the master has no data to transmit, a path test message is used to maintain the virtual circuit and also enable data transmission from the slave to the master.

The LAT system usually operates in an unbalanced mode where the terminal server is the master and initiates all communications. One exception, however, is the case of a printer. Here, the printer is the slave and the host is the master. In this situation, the printer can initiate a connection and also initiate communications when necessary.

Flow control in LAT takes place at several levels. Both a credit system and a window–based flow control system are provided at different levels. There is also an X– on/X–off system for the user as well as some flow control in the underlying data link layer.

In essence, LAT is a fairly primitive communications system that is reasonably effective in communicating with the VAX cluster. By the use of the master/slave timer system and multiplexing, LAT places a large share of the communications load on the terminal server, and thereby frees up resources on the host.

Summary

In summary, DECnet V, like OSI, represents a philosophy as much as a communications system. It incorporates much of the architecture espoused by OSI. At the same time, it provides backwards compatibility with many features of the earlier DECnet systems. In many instances, DECnet has moved in to fill gaps in the OSI architecture with proprietary protocols. These protocols are usually coordinated with other interested parties in an attempt to gain widespread acceptance of these protocols.

All in all, DECnet is a flexible, expandable network that should serve the needs of a wide variety of users. With the impending inclusion of FDDI, it should serve well into the next decade. It is a worthwhile addition to the ASRM network system.

APPENDIX

A5. GLOSSARY

ADI	Asynchronous Data Interface
API	Application Program Interface – Provided by RUST
ASC	Area Supervisory Computers
AUI	Access Unit Interface
BAS	Building Automation System
BIS	Business Information System
B_1000	Building 1000
BSGW	BASEstar Gateway
CAD	Computer Aided Design
CCTV	Closed Circuit Television
DAS	Device Access Software
DCS	Device Control Software
DHI	Data Highway Interface
FALS	Fire And Line Safety
FDDI	Fiber Distributed Data Interface
FDLM	Fiber Distributed Line Module
FTP	File Transfer Protocol
GKS	Graphics Kernel System
LSC	Local Supervisory Computers
OIS	Operational Information System
NCT	Network Cooperating Task
NI	Network Interface
PAA	Process Automation Application
PAM	Process Automation Module

PE	Protocol Emulator
PSCNI	Packet–Switched Communication Network–Internet
SR	Security Router
TCP/IP	Transmission Control Protocol/Internet Protocol
UPS	Uninterrupted Power Supply
UTP	Unshielded Twisted Pair
WS	Work–Stream

REFERENCES

1. Meeting at RUST, Birmingham, Alabama on 08/25/92 attended by Dale Scruggs (RUST), Kirk Weiss (RUST), Walter Robinson (NASA), Robert Moorhead (MSU), and Ravi Nirgudkar (MSU).

2. Handouts related to the workcell connectivity received at the 08/25/92 meeting.

3. FAX received from Walter Robinson at NASA, on 09/01/92.

4. FAX received from Kyle Maus at ASRM, on 10/05/92.

5. FAX received from Kirk Weiss at RUST, on 10/14/92.

6. Teleconference, on 10/15/92 attended by John Donaldsen (LMSC), Merlin Hill (AAD), Dale Scruggs (RUST), Kirk Weiss (RUST), Walter Robinson (NASA), Robert Moorhead (MSU), and Ravi Nirgudkar (MSU).

7. FAX received from Dale Scruggs at RUST, on 11/24/92.

8. FAX received from Merlin Hill at AAD, on 12/04/92.

9. Dale R. Thompson, "ASRM Communications Network Analysis", MS Thesis, Electrical Engineering Department, Mississippi State, Mississippi, May 92.

10. BONeS LAN Model Library Guide, USA : Comdisco Systems, Inc., 1992.

11 Multi Media Access Center Products Catalog, Cabletron Systems, Inc., 1992.

12. William Stallings, Data and Computer Communications, Macmillan Publishing Company, New York, 3rd edition, 1991.

13. Jeremiah F. Hayes, Modeling and Analysis of Computer Communication Networks, Plenum Press, New York, 1986.

14. Uyless Black, Computer Networks, Prentice-Hall, New Jersey, 1987.

15. CentreCOM Micro Transceiver Series specification sheet, Allied Telesis, Inc., 1991.

16. CentreCOM Micro 125/126 Series specification sheet, Allied Telesis, Inc., 1991.

17. CentreCOM Micro 5000 Series specification sheet, Allied Telesis, Inc., 1991.

18. Carl, Malamud, Analyzing DECnet/OSI Phase V, Van Nostrand Reinhold, New York, 1991.

19. Robert Moorhead, Wayne Smith, Dale Thompson, "An Investigation of Networking Techniques for the ASRM Facility", Final Report (first year), NASA Grant NAG8-866, February 4, 1992.

20. David Simpson, "3 Ways to Implement X", Systems Integration, Sept. 1991.

21. Ian Bacon, "Opening the X Window", MacUser, June 1992.

22. Mark L. Van Name, Bill Catchings, "AppleTalk Phase 2 and You", BYTE, Jan. 1990.

23. Dave Kosiour, Kee Nethery, "Network Speed Trials", Macworld, Dec. 1989.

24. Dave Kosiour, "Network Connections", Macworld, Nov. 1989.

25. Dave Kosiour, "On the Ethernet Highway", Macworld, March 1990.

26. Dave Kosiour, "Moving up to Ethernet", Macworld, May 1992.

27. "Ethernet Interface Tests by MacUser Lab", MacUser, June 1990.

28. Teleconference, on 01/19/93, attended by Walter Robinson (NASA), Doug Thomas (NASA), Phil Kelley (LMSC), John Donaldson (LMSC), Ted Roberts (LMSC), Dale Scruggs (RUST), Tom Reed (RUST), Robert Moorhead (MSU), Wayne Smith (MSU), and Ravi Nirgudkar (MSU).