UNIVERSITY OF WATERLOO
UNIVERSITY OF WATERLOO
UNIVERSITY OF WATERLOO
COMPUTER SCIENCE DEPARTMENT
COMPUTER SCIENCE DEPARTMENT
COMPUTER SCIENCE DEPARTMENT

A Network Operating System For Interconnected LANS With Heterogeneous Data-Link Layers

D. D. Cowan
T. M. Stepien
R. G. Veitch

Research Report
CS-87-50

September, 1987

A Network Operating System

For Interconnected LANS

With Heterogeneous Data-Link Layers

D.D. Cowan, T.M. Stepien, R.G. Veitch

Computer Science Department
and
Computer Systems Group
University of Waterloo
Waterloo Ontario Canada

September 1987

DRAFT

ABSTRACT

Microcomputer workstations interconnected by local area networks (LANs) are used extensively for educational computing in most disciplines at the University of Waterloo. The first of these LANs called Waterloo JANET was installed in 1981, and there are now over 30 such LANs in various academic areas on the campus.

Project ARIES was initiated in 1984 to extend this concept by investigating the use of portable microcomputers as fully functional portable workstations. The Project ARIES Network also uses a derivative of the Waterloo JANET LAN to support portable workstations.

In order to rationalize the use of these and other computing resources across the campus, it has become necessary to interconnect all the LANs into an Extended LAN. Because of the heterogeneity of the communication mechanisms used in the different versions of the LANs, interconnection necessitated the development of a network operating system.

This paper starts by describing the user's view of both the Waterloo JANET LAN and the Project ARIES Network. This is followed by a description of the method of interconnection of the LANs, the addressing conventions used and the implementation of the network operating system.

Research described in this paper has been supported by the University Research Incentive Fund of the Ontario Government, the National Science and Engineering Research Council of Canada, Bell Canada, Hewlett Packard (Canada), IBM Canada and the WATCOM Group.

INTRODUCTION

Faculty and staff members at the University of Waterloo have been developing and using computer systems to support education since the early 1960s. These activities have included the development of both application software and computer systems to make this software accessible to the student body. Such systems are often called delivery systems since they "deliver" useable computing power to the students.

A decision was made in 1979 at the University of Waterloo to use microcomputer workstations interconnected by local area networks (LANs) as the next generation of delivery systems for educational computing. The first such system (called Waterloo JANET[1]) became operational in 1981 and there are now over 30 around the campus serving various academic areas.

Each Waterloo JANET LAN interconnects a number of microcomputer workstations (typically 30), with file, print and communication servers. The servers provide access for each user to application software, files, print devices and other networks. These distinct delivery systems or Component LANs are also being interconnected with bridges and gateways into an Extended LAN so that students can access their own files and other materials from any workstation location on campus.

Project ARIES was started in 1984 to extend the Waterloo JANET concept by investigating the use of lap portable computers as fully functional portable workstations. Currently about 350 students are being issued with lap portable computers which have at least 512K of memory and can be carried easily between classes. Students can connect their portable computers to Transaction Machines on a campus-wide LAN, and copy files, print files, send mail and obtain other type of computing services. This system provides them with software, data, file storage, access to output devices, and specialized computing services. After having copied new data, mail and assignments from the Transaction Machine into the portable computer, the student disconnects the computer and then can perform computing activities at any convenient time and location. The campus-wide LAN serving Project ARIES is also being connected to the Extended LAN to allow students the freedom to access a wide-variety of computing resources.

A more complete description of both Waterloo JANET and Project ARIES is presented in [4] and [5].

There are a number of different Data-Link Layers used in the LANs on campus as they contain varying numbers of workstations and were installed at different times. In order to interconnect these LANs into an Extended LAN it was necessary to develop a network operating system similar to the model defined by Tripathi et al. in [18]. This network operating system would allow transparent communication over the Extended LAN and yet cope with the many different Data-Link Layers and their interconnection.

This paper presents the user's view of both the Waterloo JANET LAN and the Project ARIES Network in order to motivate the underlying LAN structure. This material is followed by a description of the method of interconnection of the LANs, the addressing conventions used, and the implementation of the network operating system.

---

[1] The name of this family of networks is an acronym for Just Another NETwork.

THE COMPONENT LANS - THE USER VIEW

The Component LANs in the Extended LAN are Waterloo JANET and the Project ARIES Network. These two Component LANs provide different types of services but they need to be connected so that users may work interchangeably between the two systems. Both LANs use the same underlying network structure and implement a service-request model similar to the one described by Summers in [15], but the two networks present quite different user interfaces. The next two sub-sections present a brief overview of Waterloo JANET and the Project ARIES Network from the user perspective.

Waterloo JANET

Workstations on Waterloo JANET are usually IBM Personal Computers or PC Compatibles with at least 256K bytes of memory and the operating system is MS or PC/DOS; workstations may also have diskette drives and a hard disk. When a user switches from a stand-alone micro-computer workstation to a network workstation, the extra learning required should be minimal; Waterloo JANET was designed with this philosophy in mind.

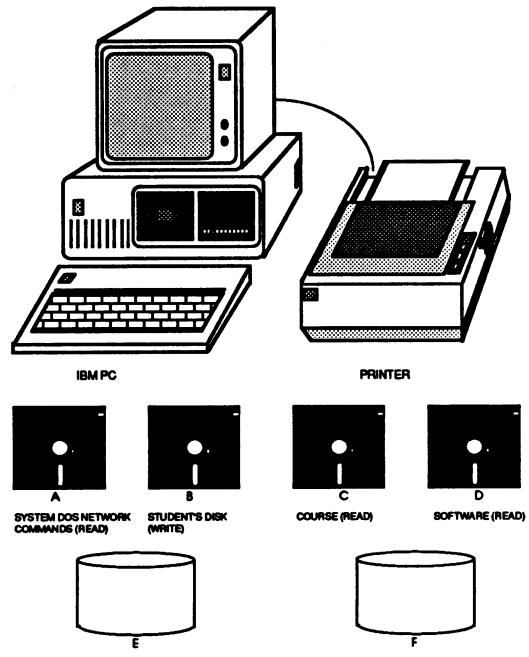

Figure 1. A User View of Waterloo JANET

When a user first sees a microcomputer workstation connected to Waterloo JANET, there is a logo and a line requesting a userid; when the userid is typed a password is requested. Once this sign-on procedure is complete, the Waterloo JANET workstation appears to the user as a normal stand-alone microcomputer with a large number (usually 4 to 8) of fixed-capacity network disks. These network disks behave as if they were diskette drives; for example, on an IBM PC the network disks are labelled A,B,C,D,E, F etc., and each can store files in a tree-structured file system. Each network disk can be a different size ranging from 20K bytes to 2.0M bytes, the size depends upon some initial allocation. Of course each network disk is actually a portion of a hard disk which is attached to the file-server. The network disks are often segregated by function, for example, there is usually at least one network disk for system software, one for application software, one for course-related materials and one for user files. A typical configuration as seen by a user is depicted in Figure 1 on page 3.

Security is extremely important when file space is shared among many users in a computer system designed for educational applications. Access to files is controlled by the userid and password required when a user signs on to the system. Also it is possible to control the amount of sharing that occurs among the different user-groups.

Almost everything else on the workstation is the same as if it were operating in standalone mode; the operating system, its commands and all the software are still available in the network environment. Some extra commands have been added to provide the Waterloo JANET environment, but most of them do not need to be learned for "normal" operation of the workstation. The extra commands in Waterloo JANET allow access to other network disks on the file-server and explicit control of the print-server.

The Project ARIES Network

The Project ARIES Network was conceived to allow users of portable computers to obtain a number of computing services including: obtaining software and data for their portable, communicating with other users, accessing printers and other output services, accessing mainframes and other networks, accessing other computing services such as high-quality text processing.

The Project ARIES Network consists of Transaction Machines, file servers, print servers and servers assigned to tasks such as archiving. Users see only a Transaction Machine which is a microcomputer with a keyboard, a screen, a printer, a system unit containing 512K bytes of memory, an Intel 80286 CPU and both 5.25 and 3.5 inch diskette drives. As well the system unit of the Transaction Machine contains adapter cards for a network connection and asynchronous communications.

To obtain services users may connect their portable computers to a Transaction Machine or may insert a diskette into the diskette drive. After connecting the portable computer or inserting the diskette the users authenticate their identity through a normal logon procedure, and this provides access to a specific user's file space. Once access to the file space and the system is permitted, the user is presented with a series of menus which make other services available. The user is allowed to operate on files on both the network file-space and the portable computer or diskette and perform the following actions: examine directories, copy files, send mail, print files, and obtain services from other servers in the network.

Selection of any one of the services described in the previous paragraph causes further menus to be presented which delineate the service even further.

WATERLOO JANET - A STRUCTURAL VIEW

The LAN being described in this paper is really a set of interconnected LANs which we call an Extended LAN. The Extended LAN is described in two stages; first we describe one of the component LANs and then describe how they are interconnected.

Waterloo JANET - A Component LAN

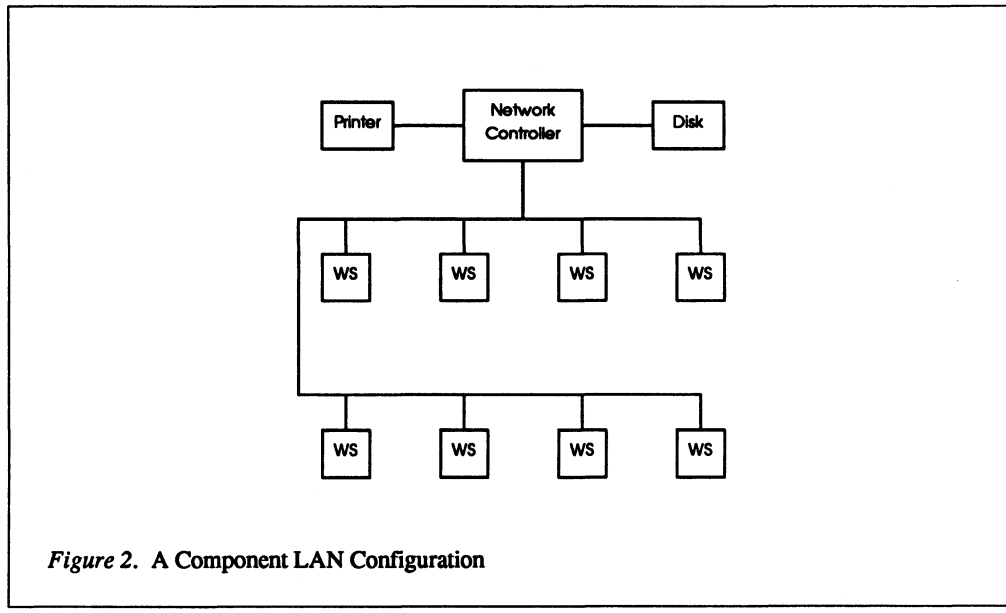

Figure 2. A Component LAN Configuration

A Component LAN consists of a number of microcomputer workstations, and file, print and communication servers interconnected by a single communications mechanism. A diagram of a typical Component LAN is shown in Figure 2.

The servers provide access for each user to both systems and applications software, files, print devices and access to other networks. Each server may be on a distinct computer or several different servers may be on a single machine. In fact most configurations of the network usually have the file, print and communication servers on one computer called the network controller (see Figure 2), as this usually provides satisfactory performance with fewer machines.

There are a number of communications mechanisms currently installed in the Component LANs at Waterloo including PC Cluster (CSMA/CA), PC Network (IEEE 802.3 and 802.2), Token Ring (IEEE 802.5 and 802.2), IEEE 488, and the IBM PC Network Baseband [8], [13], [12], [14], [10], [11], [6], [9].

Since new communications technologies are constantly becoming available, and since it is not clear which ones will predominate, the Component LAN is designed so that different communication mechanisms can be used.

Although the Project ARIES Network was designed for portable computers it has the same basic communications structure as Waterloo JANET and can also be treated as a Component LAN.

The Extended LAN

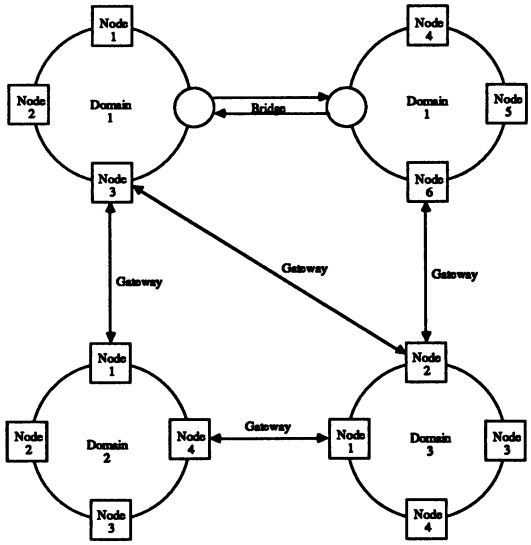

Figure 3. An Extended LAN Configuration

An Extended LAN is created by connecting Component LANs together using bridges and gateways. Bridges connect together Component LANs whose communication mechanism obeys the IEEE 802 standard, but bridges do not exist for many of the combinations which are and will be supported. In those cases special servers configured as gateways are used to interconnect two or more Component LANs. Hence it is necessary to construct a network operating system which can address the many variations in communication mechanism and methods of interconnection.

A diagram of a typical Extended LAN structure is shown in Figure 3. The Component LANs are represented by the large circles and are called domains, and each workstation, server or gateway on a domain is represented by a square and is called a node. The double-headed arrow labelled "gateway" indicates that the two nodes at either end of the gateway are really the same machine joining two different domains. The two single-headed arrows labelled "bridge" which connect the small circles, join two domains and effectively form a single domain.

Each domain is assigned a unique identifier, and each node is also assigned an identifier which is unique over a domain. A machine on more than one domain has a node identifier for each domain on which the

machine resides. Thus, workstations and servers usually have a single node identifier whereas gateways always have more than one. Each workstation and server has a unique name consisting of a domain-node pair. Of course, a gateway has more than one name because it connects two or more Component LANs.

The domains and nodes are numbered in Figure 3 on page 6 to illustrate the naming convention described in the previous paragraph. The two domains at the top of the Figure are both labelled with a one (1) since they are connected by a bridge and are the same domain. The workstation labelled three (3) on domain three (3) has the name (3,3) and the gateway labelled two (2) shown at the top of the circle representing domain three (3) has four names (3,2), (1,3), (1,6) and (2,1). The gateway has four names since it is a single machine shared by three domains and appearing twice on domain 1.

A LAN IN TERMS OF THE OSI MODEL

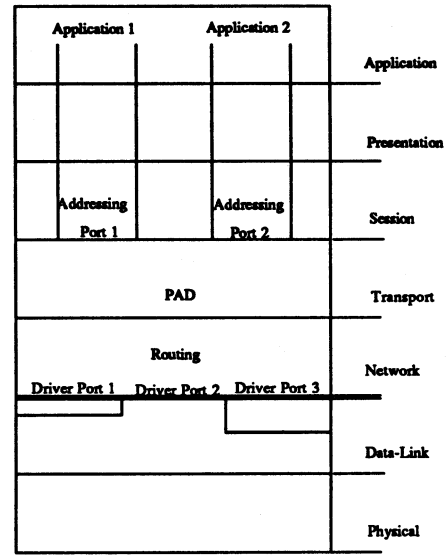

Figure 4. A LAN using the OSI Model

The Waterloo JANET LAN can be described conveniently in terms of the OSI model [16], [19], as shown in Figure 4. Applications in the Application Layer create messages which are passed through the Presentation Layer (which is null) to the Session Layer where an address is appended to the message.

Each application has a single port or socket which forms the interface into the Transport Layer. The Transport Layer implements packet assembly and disassembly for messages and also handles retransmission and acknowledgement of packets.

The Network Layer uses a routing table which contains the next step in the route to a destination domain. The routing table contains an entry for each domain in the Extended LAN. Corresponding to each domain entry in the routing table is the address of the Driver Port (which is an interface to the Data-Link Layer) and gateway which will move the packet closer to its destination. The routing table can be changed over time to compensate for any outages on the network.

Once the next step in the route has been determined in the Network Layer the packet is forwarded to a process called a Driver Port. A Driver Port provides a uniform interface between the Network Layer and the Data-Link Layer and may extend into the Data-Link Layer. A single workstation or file server would usually have one such Driver Port, while a gateway would have one Driver Port for each connected Component LAN.

The Driver Port uses the header information to modify the packet into a format suitable for transmission over a specific Data-Link Layer. Driver Ports also put a packet into a standard internal format after the packet is received. The Driver Port is the process which must be written when a different Data-Link Layer is installed.

The diagram in Figure 4 on page 7. represents a gateway since three (3) Driver Ports are indicated. One of the Driver Ports shown as a line is strictly an interface, whereas the two other Driver Ports are shown as boxes. These Two Driver Ports are more than interfaces as they perform formatting for Data-Link Layers which do not conform to the IEEE 802 standard.

Since packets moving between Component LANs may pass through Data-Link hardware which uses different packet sizes, it was decided to handle packets in two different ways. Packets which remain in their original domain use the optimum packet size for the Data-Link hardware. Packets leaving their original domain are decomposed into the minimum packet which is compatible with all the Data-Link hardware. This design decision was made because the size of a message is not fixed and it is possible that a gateway would not have enough memory to re-assemble the message. Obviously not re-assembling packets removes the overhead associated with the Transport Layer, but at the same time it does not make optimum use of the capacity of the Data-Link Layer. The current system has a minimum packet size of 512 bytes.

ADDRESSING OF MESSAGES

The Address Format

Addressing in the entire extended LAN attempts to follow the format of the IEEE 802 standard. In this format each packet has a header which contains a 6-byte Destination Address, a 6-byte Source Address, and a Route. This form of address assumes that every device on the Extended LAN is uniquely identified by a single identifier and that the Extended LAN is a single domain. The information portion of the packet may also contain Destination Service Access Point (DSAP) and Source Service Access Point (SSAP) fields which correspond to the port or socket shown in Figure 4 on page 7.

This addressing scheme will operate correctly if all Data-Link Layers obey the IEEE 802 standard, but there are still many Data-Link Layers which do not implement the standard and yet need to be incorporated into the Extended LAN. The extension of the standard is achieved by introducing a three-level addressing scheme into the IEEE 802 format. Each Application Port or socket (see Figure 4 on page 7) on each Component LAN is addressed by a 1-byte domain number representing a Component LAN, a 6-byte node number representing a single computer, and a 1-byte port number which connects to an Application Port. The total addressing scheme for a message is shown in Figure 5. Both an 8-byte desti-

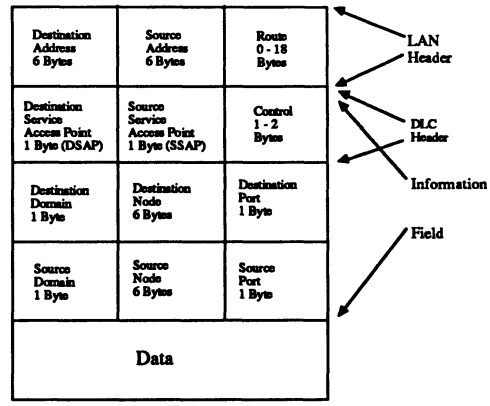

Figure 5. The Address Structure in a Message

nation address and an 8-byte source address using the domain-node-port format are placed in the information field. This extra addressing information reduces the data capacity of a 512-byte packet by about three per cent (3%).

When a packet passes through an Extended LAN which is connected by bridges using the IEEE 802 standard the extra addressing is ignored by the bridge. The extra addressing only comes into play when a packet passes through a gateway which joins two Component LANs.

The Addressing Algorithm

An application wishing to send a message to a server obtains the address of that server by broadcasting to all servers in the network. The server returns its complete address including the information required for the fields in the application message labelled Route, Destination Domain, Destination Node and Destination Port as shown in Figure 5. Since a node may be on a component LAN which does not use the 802 standard, the field labelled Destination Node may not always be in the usual 802 format. All the addressing information for the destination except the field labelled Destination Address in the LAN Header in Figure 5, is filled in by the Session Layer, although the Session Layer is only completing a template and does not know the internal structure of the address. When the message is divided into packets in the Transport Layer the entire address header is appended to each packet.

The Network Layer fills in the complete source address including the fields labelled Source Address, Source Domain, Source Node and Source Port. Using the routing table in the Network Layer the Destination Address field in the LAN Header is completed and the packet is directed to the correct Driver Port.

The rest of this section describes the algorithm used to fill in the Destination Address field. An Extended LAN with the driver ports labelled with letters is shown in Figure 6. Workstations and servers are

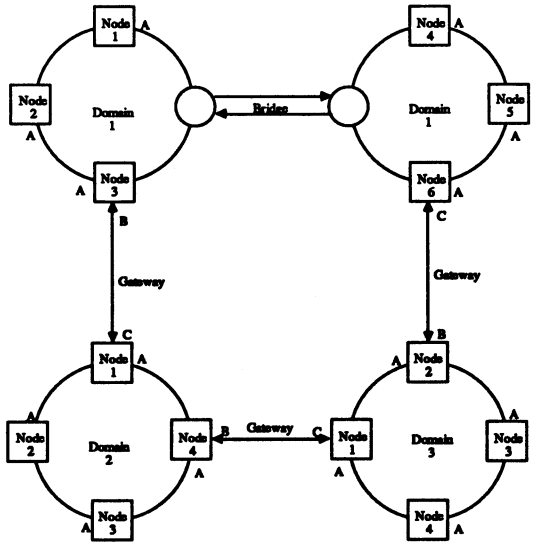

Figure 6. An Extended LAN Showing the Driver Ports

Routing Table for Node (1,1)

Domain	Driver Port	Gateway
1	A	Null
2	A	3
3	A	3

Routing Table for Node (1,3)

Domain	Driver Port	Gateway
1	A	Null
2	C	Null
3	C	4

Routing Table for Node (1,5)

Domain	Driver Port	Gateway
1	A	Null
2	A	3
3	A	3

Figure 7. Routing Tables for A Workstation or Server, a Gateway and a Workstation Using a Bridge

labelled with a single letter indicating one Driver Port while gateways have at least two letters indicating a Driver Port for each connected Component LAN. Three possible cases can occur when the packet is routed by the Network Layer. These three cases depend on whether a packet is about to pass through a workstation or server, a gateway, or a bridge, and are illustrated by the routing tables in Figure 7 on page 10. Each case is discussed in the next three paragraphs

The case of a workstation or a server is represented by the top routing table in Figure 7 on page 10. If the packet is to remain in the current domain then the contents of the Destination Node field is copied to the Destination Address field. If the packet is meant for another domain then the address of the gateway from the Gateway field in the routing table is copied into the Destination Address field. The packet is then sent to Driver Port A for distribution over the network.

The routing table for a gateway which is connected to two Component LANs is the middle table in Figure 7 on page 10. If the packet is destined for domain two (2) which is the adjacent domain then the Gateway entry in the routing table contains a Null entry. In this case the contents of the Destination Node field is copied to the Destination Address field. If the packet is meant for another domain then the address of the gateway from the Gateway field in the routing table is copied into the Destination Address field. The packet is then sent to Driver Port C for distribution over the network.

A bridge connects together two or more Component LANs which obey the 802 standard, and makes the Component LANs into a single domain. The routing table for Node (1,5) which is a node on this type of network is shown as the last entry in Figure 7 on page 10. If the packet is to remain in the domain the Destination Node field is copied to the Destination Address field, and the packet is then sent to Driver Port A for distribution over the network. The bridge would obtain its information from the Route field in the LAN Header.

IMPLEMENTATION OF THE LAYERS

Each layer is composed of a set of processes which communicate by passing messages using a simple set of primitives. The message-passing model is similar to the one described by Gentleman [7], and used in Thoth [1], [3], MINIX [17], and V Kernel [2]. Messages can be either between processes in the same machine or different machines.

The process mechanism is implemented with a kernel containing a real-time scheduler which provides fast-context switching. The kernel also contains procedures for four message-passing primitives which allow implementation of interprocess communication on the same computer. The kernel is included as part of the resident operating system on each workstation, server and gateway in the Extended LAN.

Communication among processes on different computers is implemented using a queue manager which is resident in the operating system on each workstation, server and gateway. This queue manager is a process which transfers messages between Application Ports and Driver Ports in order to send them over the Extended LAN.

The next three sub-sections describe the message-passing mechanism, the real-time kernel and the queue manager.

The Message Passing Primitives

There are four message-passing primitives available which allow implementation of interprocess communication on the same computer. The four primitives are: Send, Receive, Reply and Signal and these are completely adequate as shown by Cheriton [1], [3], [2]. Messages contain the text of the message and the address of the recipient. When a message arrives at its destination the address of the recipient is replaced with the address of the sender, so that the receiver can easily reply. The next four sub-sections present a brief description of the four primitives.

The Send Primitive: The Send primitive sends a message to a receiving process, and returns a value which indicates success or failure. Failure occurs if no receiving process exists. Once the Send primitive is executed, the process sending the message is blocked (can not run) until a Reply is received. Reply is defined in a later section.

The Signal Primitive: Signal sends an unsigned integer to the named process, thus "signalling" the process and causing it to unblock.

The Receive Primitive: The process executing the Receive primitive receives a message from any other process. Receive only knows the identity of the sending process when the message arrives, as the identity of the sending process is copied into the message. The Receive primitive also has an associated time-out value. A process executing the Receive primitive blocks until a message arrives or the time-out value is exceeded.

Receive returns a value which is zero if unblocking occured because of a message arrival, and non-zero if unblocking occurred because of a time-out value being exceeded. Receive can also be unblocked by a Signal primitive which would also cause a non-zero value to be returned.

The Reply Primitive: The Reply primitive responds to a message sent by a process, and returns a value which indicates success or failure. Failure occurs if no sending process exists or if the named process is not waiting for a reply. When the Sending process receives a reply to its message, the sending process is unblocked. Reply can also return a message to the sending process.

The Kernel

The kernel is a real-time scheduler which allows fast context-switching in order to implement a process-based system. When a process is blocked by the use of a Send or Receive primitive the process's context information is placed in a blocked queue. When a process is unblocked it is placed in a queue of processes waiting to be run or dispatched on the processor. Processes are dispatched using a FIFO discipline, and processes in a higher priority queue are always dispatched first. A process will only surrender its use of the processor if the process is blocked, or if a process of a higher priority is waiting to be dispatched. By using a Receive with a timeout delay and no message, a process can be blocked and forced to surrender its control of the processor. The Signal primitive will also cause a process to be dispatched if the process has a higher priority.

There are both Spawn and Kill primitives implemented in the kernel to allow the construction and destruction of processes. Spawn creates a process from a template and returns a process identifier (pid), while the Kill primitive destroys a process.

The operating system on the workstations, servers and gateways is PC or MS/DOS and the kernel is implemented as a resident part of this system. The kernel intercepts all hardware interrupts before they reach the operating system and performs the dispatching function before passing control to the operating

system. Processes can also be dispatched whenever a message-passing primitive is executed as these are implemented as procedure calls to the kernel.

All Input/Output and other system functions are provided through calls to the resident operating system. Since MS/DOS is serially re-useable this can cause problems if a process must give up control during a system call. The operating system must be treated as a critical section but it is possible with the limited set of primitives to prevent a problem from occurring.

The kernel is quite small, it adds 8K bytes to the resident portion of the operating system and requires 40 bytes to queue each process when it is not running. Performance measurements on the kernel indicate that a context-switch takes 250 microseconds on an IBM PC-AT with a 8 mHZ clock, and 45 microseconds to send, receive or reply with a message of 10 bytes.

The Queue Manager

One of the processes running on each workstation, server and gateway in the Extended LAN is a Queue Manager called "Switch". The process Switch accepts a message from a sending process and delivers the message to a receiving process. The message must contain the domain, node and port address for both sending and receiving processes as described earlier in this paper. With this information Switch can deliver a message between processes residing on the same or different machines; however Switch is primarily used for intermachine communication.

Each process which uses Switch is assigned a reception queue. Any message sent by a process is transferred by Switch to the reception queue of the receiving process. Thus Switch implements a general-purpose Courier process [7], which allows asynchronous buffered communication between processes.

When Switch is used for intermachine communication the message is placed in the reception queue of a Driver Port. The Driver Port is chosen by consulting the routing table contained in the Network Layer.

CONCLUSIONS

This paper has described the structure of Waterloo JANET a Component LAN and a large Extended LAN which was created by joining the Component LANs with gateways and bridges. Because of the variation in Data-Link Layers, many of which did not follow the current standards it was necessary to construct a network operating system which would tolerate their differences.

This network operating system was implemented using the message-passing paradigm. An existing operating system was modified so that it would support processes and message-passing primitives between processes which could be on the same or different machines.

A second form of addressing was introduced which was incorporated into the IEEE 802 packet format without significantly altering its data-carrying capacity. With this extra addressing capability it was possible to handle all the different types of Data-Link Layers currently used in the Extended LAN.

Preliminary experience with the network operating system and the extended addressing capability indicates a level of performance which is quite acceptable.

BIBLIOGRAPHY

1. Cheriton, D.R. *Multi-Process Structuring and the Thoth Operating System* Waterloo Ontario: Computer Science Department University of Waterloo, 1979.

2. Cheriton, D.R. *The V Kernel: A Software Base for Distributed Systems* IEEE Software, April 1984, pp 19-42.

3. Cheriton, D.R. Malcolm, M.A. Melen, L.S. and Sager, G.R. *Thoth, A Portable Real-Time Operating System* Communications of the ACM Vol. 22, No. 2, February 1979, pp 105-115.

4. Cowan D.D., Fenton S., Graham J.W., Stepien T.M. *Networks for Education at the University of Waterloo*, Computer Systems Group University of Waterloo, Waterloo Ontario Canada: April 1987. "Submitted to Computer Networks and ISDN Systems"

5. Cowan D.D., Stepien T.M. *Project ARIES A Network for Convenient Computing in Education*, Computer Systems Group University of Waterloo, Waterloo Ontario Canada: July 1987. "Submitted to the 1988 International Seminar on Digital Communications"

6. Fisher, E. and Jensen, C.W. *PET and the IEEE 488 Bus (GPIB)*. Berkeley California: OSBORNE/McGraw-Hill, 1980 ISBN 0-931988-31-4

7. Gentleman, W.M. *Message Passing Between Sequential Processes: The Reply Primitive and the Administrator Concept* Software Practice and Experience, Vol. 11 August 1981, pp 435-466.

8. IBM *Personal Computer Hardware Reference Library Technical Reference Options and Adapters Volume 2*. Boca Raton Florida: IBM Corp., 1984

9. IBM *Personal Computer Hardware Reference Library PC Network Baseband Adapter Technical Reference*. IBM Corp., 1987

10. IBM *Personal Computer Hardware Reference Library Technical Reference Token-Ring Network PC Adapter*. IBM Corp., 1986

11. IBM *Token-Ring Network Architecture Reference*. IBM Corp., 1986

12. IEEE *ANSI/IEEE Std 802.2-1985 ISO/DIS 8802/2*. 345 East 47th Street New York NY 10017: IEEE, 1985 ISBN 0-471-82748-7

13. IEEE *ANSI/IEEE Std 802.3-1985 ISO/DP 8802/3*. 345 East 47th Street New York NY 10017: IEEE, 1985 ISBN 0-471827-49-5

14. IEEE *ANSI/IEEE Std 802.5-1985 ISO/DP 8802/5*. 345 East 47th Street New York NY 10017: IEEE, 1985 ISBN 0-471829-96-X

15. Summers, R.C. *A Resource Sharing System for Personal Computers in a LAN: Concepts, Design, and Experience* IEEE Transactions on Software Engineering, Vol. SE-13 No. 8 August 1987, pp 895-904.

16. Tanenbaum, Andrew S. *Computer Networks*. Englewood Cliffs, New Jersey: Prentice Hall, 1981 ISBN 0-131651-83-8

17. Tanenbaum, Andrew S. *Operating Systems Design and Implementation.* Englewood Cliffs, New Jersey: Prentice Hall, 1987 ISBN 0-13-637406-9

18. Tripathi, S.K., Huang, Y., and Jajodia, S. *Local Area Networks: Software and Related Issues* IEEE Transactions on Software Engineering, Vol. SE-13 No. 8 August 1987, pp 872-879.

19. Voelcker, J. *Helping computers communicate.* IEEE Spectrum March 1986, pp61-70, IEEE, 1986